



THE BITBUS™ INTERCONNECT SERIAL CONTROL BUS SPECIFICATION

In the United States, additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the **implied** warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no **responsibility** for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no **responsibility** for the use of any circuitry other than circuitry embodied in an Intel product. No other **circuit** patent licenses are implied.

Intel software products are copyrighted by and shall remain the **property** of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's Software License Agreement, or in the case of software delivered to the government, in accordance with the software license agreement as defined in FAR 52.227-7013.

No part of this document may be copied or reproduced in any form or by any means without **prior** written consent of Intel Corporation.

Intel Corporation retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The **following** are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

Above	iLBX	Intellink	MICROMAINFRAME	Ripplemode
BITBUS	im	iOSP	MULTIBUS	RMX/80
COMMPuter	iMDDX	iPAT	MULTICHANNEL	RUPI
CREDIT	iMMX	iPDS	MULTIMODULE	Seamless
Data Pipeline	Inboard	iPSC	ONCE	SLD
ETOX	Insite	iRMK	OpenNET	SugarCube
FASTPATH	Intel	iRMX	OTF	UPI
Genius	intel	iSBC	PC BUBBLE	VLSiCEL
Δ	Intel376	iSBX	Plug-A-Bubble	376
I	Intel386	iSDM	PROMPT	386
i ² ICE	intelBOS	iSXM	Promware	386SX
ICE	Intel Certified	KEPROM	QueX	4-SITE
iCEL	Intelevision	Library Manager	QUEST	
iCS	Intelligent Identifier	MAPNET	Quick-Erase	
iDBP	Intelligent Programming	MCS	Quick-Pulse Programming	
iDIS	Intellec	Megarhaaia		

REV.	REVISION HISTORY	PRINT DATE
A	Original Issue	1184
B	Minor Revisions to Original Issue	7188

1.0 GENERAL

1.1 SCOPE

Connection of dedicated controllers and process I/O points to a control computer has been a long-time problem. Traditional methods have used a standard parallel bus to interconnect I/O expansion boards, which provide the required signal conditioning. In many cases, these buses are satisfactory; however, in many other cases the characteristics of these parallel buses create significant limitations. Electrically, parallel buses limit the number of boards that can be transparently (i.e. in the same backplane) added to the system. Mechanical packaging for parallel buses often makes I/O cabling cumbersome, due to the close proximity of boards within a backplane. Most importantly, the fixed form factor and bus interface logic requirements of a parallel bus put a lower bound on future system cost reduction, regardless of the advancing capabilities of future VLSI. The BITBUS™ interconnect is a serial control bus specifically designed to address these problems.

The BITBUS interconnect allows up to 250 nodes to be easily interconnected over a physically distributed domain. Distribution capability ranges from 30 meters for the synchronous mode of operation to thousands of meters for the self clocked mode of operation. The different modes of operation are optimized for a wide range of applications covering the spectrum from high speed servo motor control in robotics, to long distance environmental control. In all cases the BITBUS interconnect is optimized for the high speed transfer of short control messages in a hierarchical system.

The level of specification for the BITBUS interconnect is somewhat different from traditional parallel bus structures. In the past, serial connections have been slow, software intensive, and hard to use. A goal of the BITBUS interconnect is to provide an easy to use, high performance serial interconnect that is transparent to the application programmer. For this purpose, this specification not only defines electrical and data link protocol aspects of the bus, as is typical for parallel buses, but also specifies a message structure and protocol for a multitasking environment, and a set of high level commands for remote I/O access and application task control. This allows standard high level software interfaces to be written, off-loading the application programmer of this complication. More importantly, this high level of specification allows the standard interface to be driven into silicon, further reducing the interface overhead, cost and complexity.

This specification has been prepared for those users intending to design or evaluate products that will be compatible with the BITBUS interconnect. The intent of this specification is to specify the interface requirements only. The implementation of interface logic and software is left to the user.

1.2 DEFINITIONS

System — A set of interconnected elements which achieve a given objective through performing a specific function. In this specification a BITBUS system consists of one master node and from one to 250 slave nodes attached to the same BITBUS interconnect.

Message-Passing — The transfer and control of structured data between two tasks.

Task — An entity which competes for system resources in order to execute a program.

Protocol — The rules by which information is exchanged across an interface.

Operation — The process whereby information is transferred between two elements across an interface.

Interface — A shared boundary between two elements within a system.

Bit-Cell-Time — The time interval required to transfer one bit of data on a serial line.

2.0 FUNCTIONAL OVERVIEW

The BITBUS interconnect concept allows highly reliable, low cost distributed expansion of process

I/O points and intelligent I/O controllers. This section provides an overall understanding of the concept by describing the hierarchical model, bus elements, modes of operation and levels of specification. Detailed specifications are provided in subsequent sections.

2.1 HIERARCHICAL MODEL

The BITBUS interconnect is defined to provide a high speed serial control bus for hierarchical systems. This model, shown in figure 1, implies that a single master must communicate with multiple slaves. These slaves may consist of simple I/O points or they may be highly intelligent controllers. By adopting this architectural model, the BITBUS interconnect is able to provide a highly capable, low cost solution, especially for those applications that match the model. This is achieved through a significant reduction in protocol complexity and overhead that would otherwise be necessary to implement a more complex multiple access type protocol. Evidence of this simplicity can be seen by comparing the level of protocol support provided in hardware for the BITBUS interconnect as compared to other more complex protocols.

The hierarchical model for the BITBUS interconnect may consist of one or multiple levels as shown in figures 1a and 1b respectively. The multiple level hierarchy is extremely useful in many applications, especially where BITBUS interconnects must run at different speeds. This specification provides the necessary hooks to implement such a structure with minimal overhead. In this specification a BITBUS system refers to only the single level hierarchy. The multiple level hierarchy consists of multiple BITBUS systems.

The goal of the BITBUS interconnect is to provide a message passing interface between tasks at the master node and tasks at multiple slave nodes within the hierarchical model. This is accomplished through an **order/reply** message protocol. Tasks on the master node issue orders to tasks on the slave nodes and tasks on the slave nodes respond with replies. This level of support effectively hides the serial nature of the BITBUS interconnect from the application programmer, resulting in a relatively simple interface.

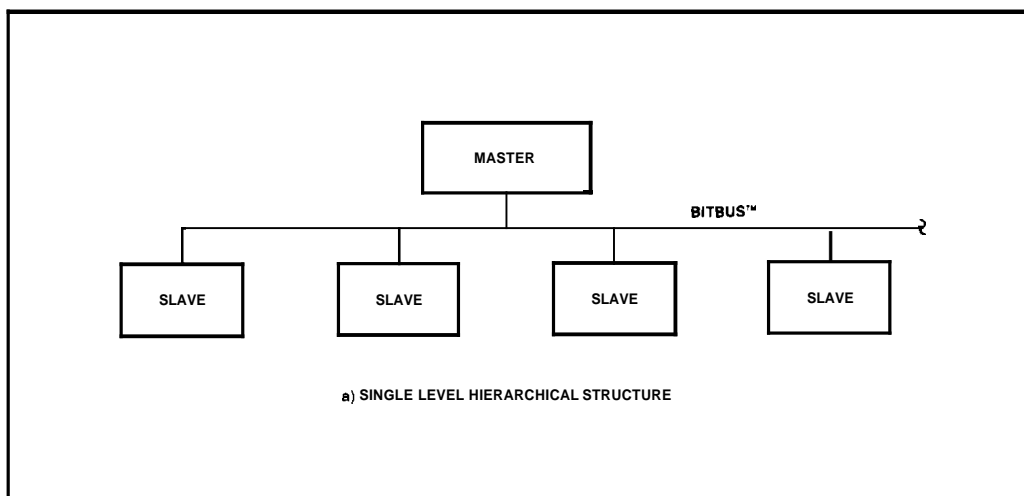


Figure 1a. BITBUS™ Interconnect Hierarchy

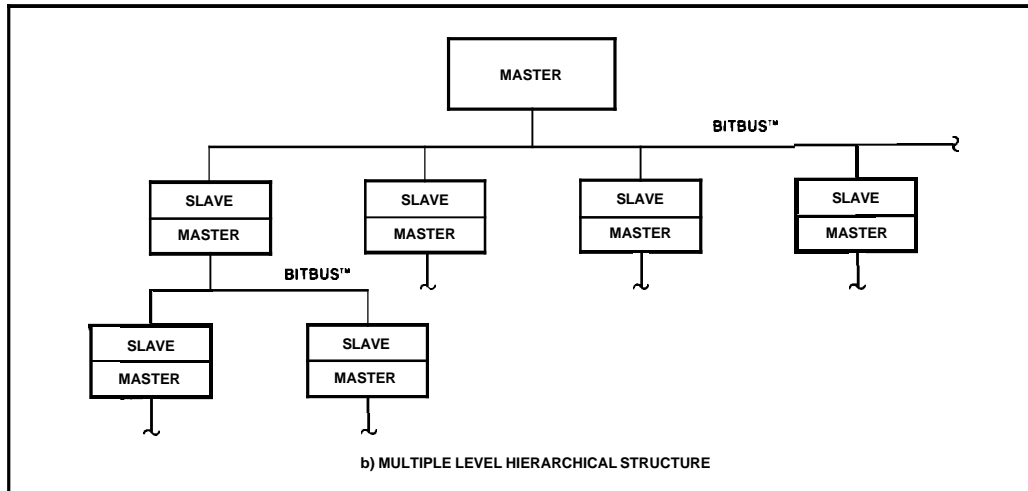


Figure 1b. BITBUS™ Interconnect Hierarchy

2.2 BUS ELEMENTS

The basic element in a **BITBUS** system is a node. A node may consist of either a device, a device and an extension, or a repeater. There are two types of devices: the master device and a slave device. This section explains the unique elements of the **BITBUS** system: the master device, the slave device, the extension, and the repeater. Figure 2 illustrates these elements.

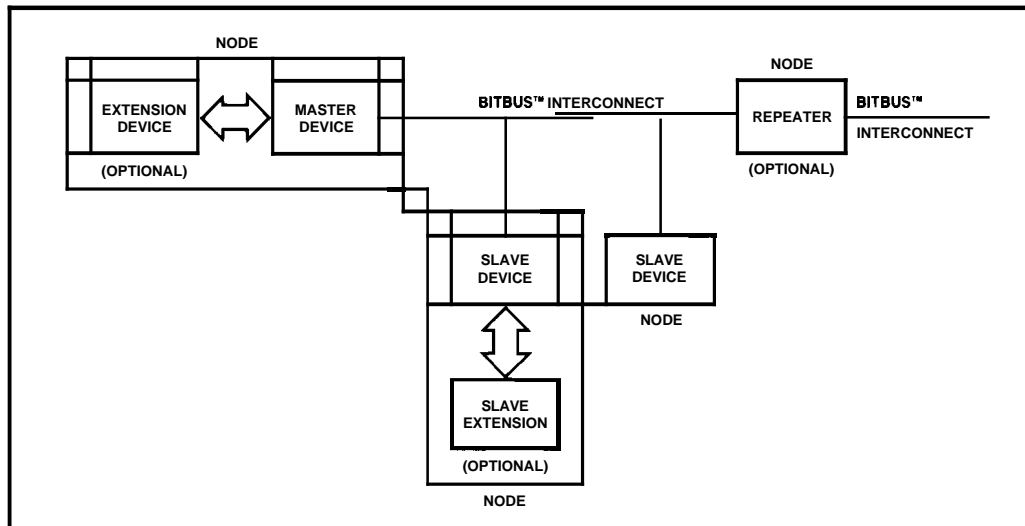



Figure 2. BITBUS™ Interconnect Elements

2.2.1 MASTER DEVICE

The master device controls all **BITBUS** interconnect operations. There is only one master device per **BITBUS** system. The master device initiates a transaction by sending an order message to a slave device. It then continues to poll the slave device until a reply message is returned.

During normal operation, each transfer from the master device is acknowledged by the addressed slave device. If a reply is not immediately available, a simple data link acknowledgement is returned.

 This frees the master device to perform other operations, such as sending another message or polling another slave device. In general, this scheme maximizes the available bus bandwidth within the BITBUS system.

The master device may be stationary in one node or this function may be passed between nodes. Mastership passing in a BITBUS system is performed via a scheme similar to token passing; however, it reduces system reliability (e.g. possibility of lost "tokens") and is therefore in general discouraged. On the other hand, despite the lower reliability, there may be specific cases where passing of mastership is desirable. One example is the implementation of a redundant master. In this case, it may be possible to increase the overall system mean time between failure (MTBF), even with the reduced reliability of passing mastership. This is especially true if all potential masters can be confined to a small section of cable isolated from the rest of the system by a repeater. Note that mastership may not be passed through repeaters.

2.2.2 SLAVE DEVICE

A slave device is a replier in a BITBUS system. There may be up to **250 slaves** per system (addresses 1-250). As a replier, **the slave** may not spontaneously initiate a transfer. Rather, it may only respond to orders from the master device. This approach significantly simplifies the slave device protocol requirements allowing powerful, low cost solutions to be implemented using highly integrated micro-controllers.

2.2.3 EXTENSION

An extension is a secondary processor within a node. The interface between a master or slave device and its extension is not part of this specification. The reason for including the extension in this description is to identify the need for a control field within the standard BITBUS message format to efficiently route messages to a master or slave device or their respective extensions. This allows low cost nodes to be constructed with a single processor and high capability nodes to be constructed with an application processor (extension) and a BITBUS interface processor. By differentiating tasks that run on these two processors, the BITBUS interface processor may contain tasks to off-load the extension processor while the extension processor maintains the capability to send messages directly through to the BITBUS interconnect. This capability may also be used to implement gateways between multiple BITBUS systems creating the multiple level hierarchy shown in Figure 1b.

2.2.4 REPEATERS

A repeater is a node used to regenerate (not **reclock**) the BITBUS interface signals. Repeaters are used to extend the distance or node count within a BITBUS system. Repeaters are referred to as nodes since they load a segment just as a master or slave node does. Repeaters are only allowed in the self clocked mode of operation. Section 2.3.2 provides more details on repeater operation.

2.3 MODES OF OPERATION

The BITBUS interconnect may operate in one of two modes: synchronous mode or self clocked mode. The purpose for multiple modes is to provide a wide range of **performance/distance** options for a variety of applications. In this section each mode is presented with its corresponding features and signal line definitions. Note that in both modes of operation all BITBUS interconnect signals are differential. The terminology used to specify the signal pairs is NAME and NAME'. State definition for the signal pairs is provided in section 3.1.1 of this specification.

2.3.1 SYNCHRONOUS MODE

The synchronous mode of operation is optimized for the highest performance over a relatively short distance. This mode can interconnect up to 28 nodes over 30 meters at transmission speeds between 500 kbits/sec and 2.4Mbits/sec. A typical system is shown in figure 3.

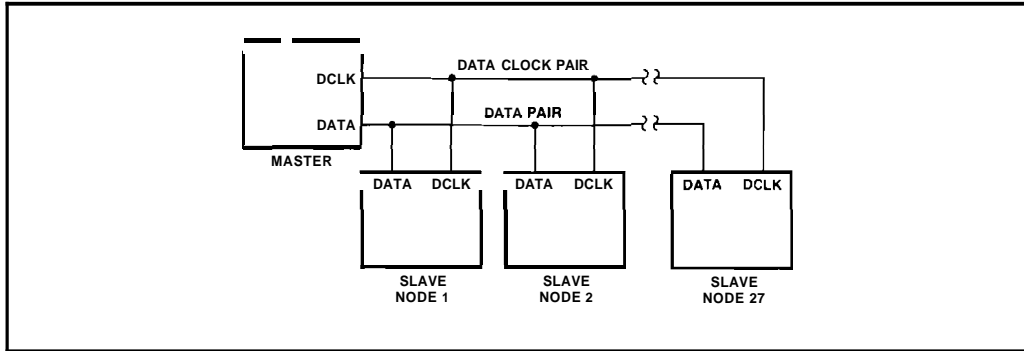


Figure 3. Synchronous Mode Connection

The synchronous mode uses two differential signal pairs: one for data (DATA, DATA *) and one for the data clock (DCLK, DCLK *). The data signal pair carries data which is referenced to the data clock signal pair. Data changes on the "falling edge" of the data clock pair and is sampled on the "rising edge". A sample interface circuit is shown in figure 4. Note that the data clock source is always at the transmitting node.

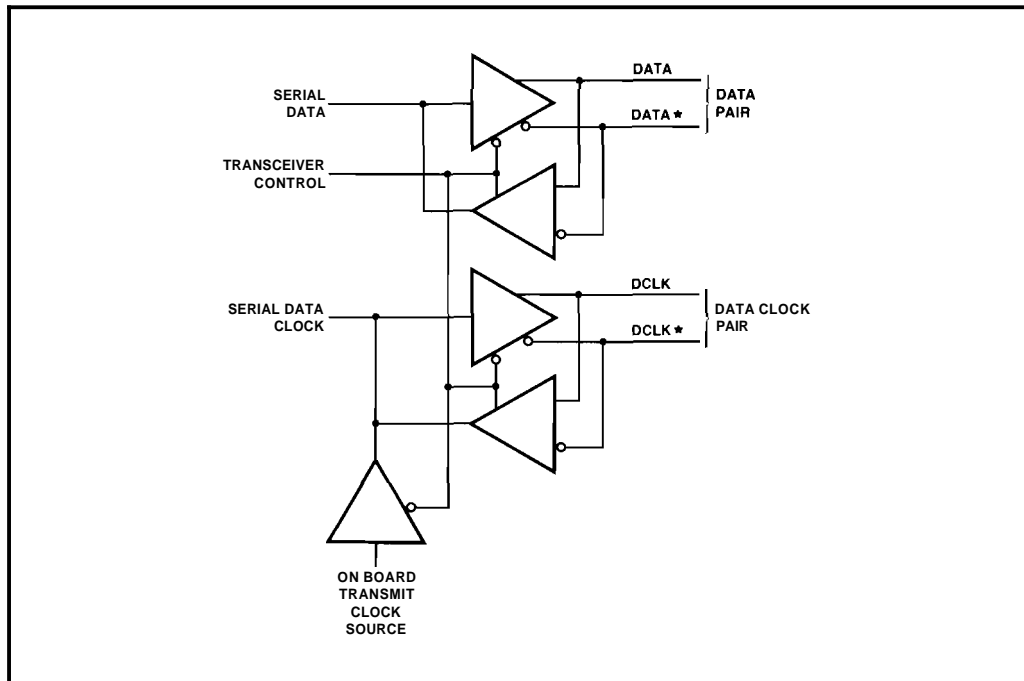


Figure 4. Typical Synchronous Mode Interface

2.3.2 SELF CLOCKED MODE

The self clocked mode of operation is provided to allow longer distance operation. There are **two**

standard bit rates for this mode of operation: **375 kbit/sec** and **62.5 kbit/sec**. The **BITBUS** interconnect supports segments up to **300 meters** in length at **375 kbit/sec** and **1200 meters** in length at **62.5 kbit/sec**. Each segment supports up to **28 nodes**. By connecting segments via repeaters, this mode supports up to **250 nodes** over several thousand meters. A typical system is shown in figure 5.

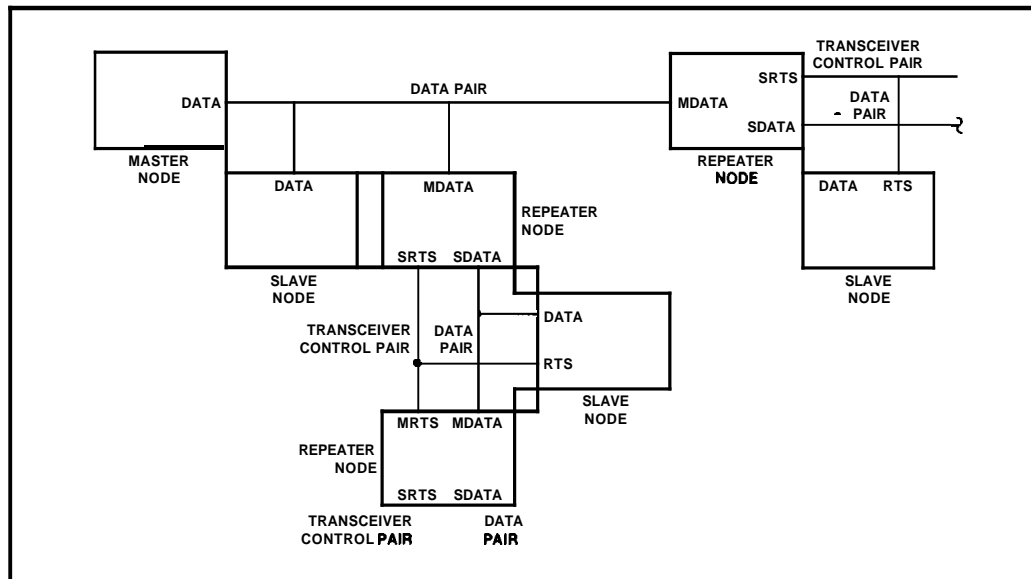


Figure 5. Self Clocked Mode Connection

The self clocked mode uses two differential signal pairs: one for data (DATA, DATA*) and one for transceiver control (RTS, RTS*). The data signal lines carry Non-Return to Zero Inverted (NRZI) encoded data. This encoding method combines clock and data onto the same signal pair. The transceiver control signal pair is used to control transceivers within the repeaters. When repeaters are not used, the transceiver control pair may be omitted. A sample interface circuit is shown in figure 6.

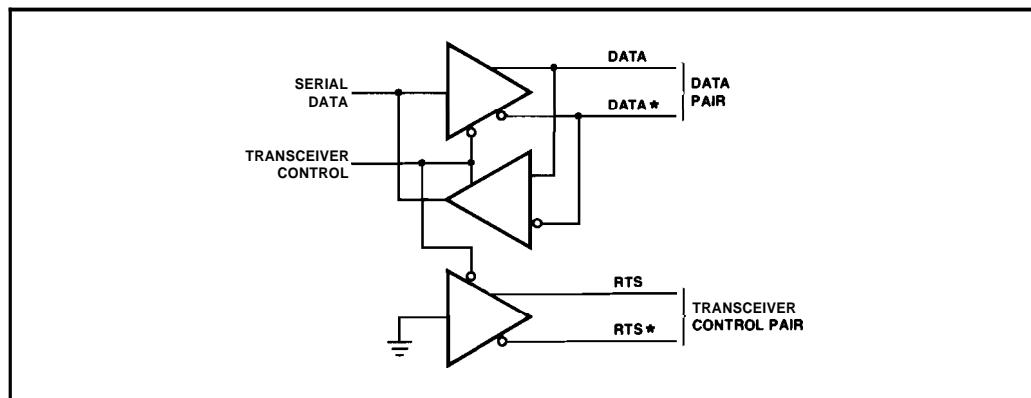


Figure 6. Typical Self Clocked Mode Interface

A BITBUS repeater may or may not provide electrical isolation, depending on the application requirements. In either case, when a slave device is not transmitting, biasing resistors enable the repeater transceivers away from the master device, allowing the master device to transmit to all slaves. When a slave device responds, it reverses the polarity of the transceiver control pair, which reverses the direction of all repeaters between it and the master device. Figure 7 shows a sample repeater circuit.

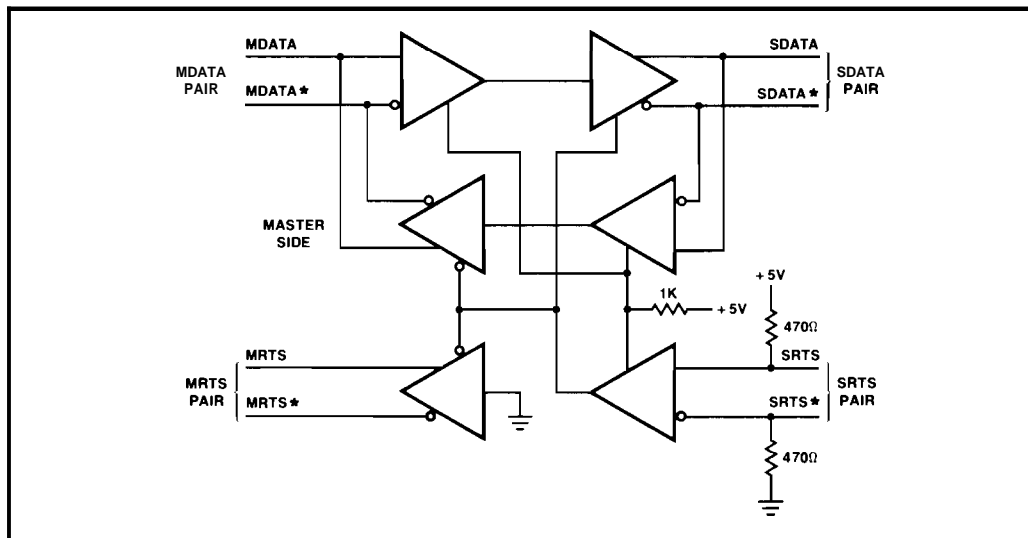


Figure 7. Typical Repeater

2.4 LEVELS OF SPECIFICATION

The BITBUS interconnect is designed to provide a highly capable, easy to use interface to the application programmer. In order to be easy to use, this interface must hide the complexity of controlling the serial link from the user through a standard **hardware/software** interface. In order to be highly capable, the BITBUS interconnect must be defined at a high enough level to allow this hardware and software interface to be efficient, and to a large degree be driven into hardware. These requirements result in the need for the following levels of specification:

- Electrical Interface
- Data Link Protocol
- Message Protocol
- Remote Access and Control
- Mechanical

2.4.1 ELECTRICAL INTERFACE

The BITBUS electrical interface specification defines the mechanisms by which bits are transferred. This section of the specification is simplified by the fact that the BITBUS interconnect consists of only one or two signal pairs and the electrical interface is based to a large degree on existing standards. This level of the specification is optimized for low cost implementations (i.e. low cost cable and low cost transceivers).

2.4.2 DATA LINK PROTOCOL

The BITBUS data link protocol specification defines the mechanisms by which packets are reliably transferred across the electrical interface. Reliability aspects include error detection, and, in many

cases, error recovery. This is important to the **BITBUS** interconnect definition since it makes the application less sensitive to the serial nature of the interconnect. This section of the specification is

also based to a large degree on existing standards.

2.4.3 MESSAGE PROTOCOL

The **BITBUS** message protocol specification defines the mechanism by which tasks pass messages over the data link. The protocol is optimized for communication between tasks on the master node and tasks on slave nodes. In actual implementations this same protocol can be used to communicate between tasks on the same node as well, allowing the serial bus to appear transparent. The purpose for providing this level in the bus specification is to allow a standard high level user interface to be efficiently implemented, and eventually driven into silicon. This section of the specification is based on a simple high performance protocol, using a well established order/reply mechanism.

2.4.4 REMOTE ACCESS AND CONTROL

The remote access and control (RAC) specification defines a set of high level commands to perform a set of standard operations over the **BITBUS** interconnect. Remote access commands allow standard I/O operations (e.g. read, write, etc.) to be performed on up to 256 I/O ports at each slave node without application software. For higher capability systems remote control commands allow slave node tasks to be downloaded, uploaded and controlled from the master device.

2.4.5 MECHANICAL

The **BITBUS** mechanical specifications are minimal. The purpose for these specifications is to define standard connectors for the **BITBUS** interconnect and an optional I/O board form factor for higher levels of compatibility.

3.0 ELECTRICAL INTERFACE SPECIFICATION

The **BITBUS** interconnect electrical specification requires definition of data encoding techniques and DC/AC parameters for the interface logic and interconnect cable. This section provides the required definition and specifically points out where existing standards are used as a basis for the specification.

3.1 DATA ENCODING TECHNIQUES

The **BITBUS** interconnect uses different data encoding techniques for the two modes of operation. Each is discussed separately below, along with a general definition of signal pair state.

3.1.1 GENERAL SIGNAL PAIR STATE DEFINITION

All signals on the **BITBUS** interconnect are differential pairs. The naming convention for these pairs identifies the two signal lines as NAME and NAME*. An active signal (logical 1) is present when the NAME signal line is at a higher electrical potential than the NAME* signal line. An inactive signal (logical 0) is present when the NAME signal line is at a lower electrical potential than the NAME* signal line.

3.1.2 SYNCHRONOUS MODE

Data encoding is relatively simple in the synchronous mode. The logical value of each bit is simply indicated by the level of the data signal pair at the bit cell center. The bit cell center is defined as the rising edge of the data clock pair. A logical zero is represented by the DATA signal being at a lower electrical potential than the DATA* signal at the bit cell center. A logical one is represented by the DATA signal being at a higher electrical potential than the DATA* signal at the bit cell center. These relationships are shown in figure 8. Data encoding for the synchronous mode of operation

must also provide zero bit insertion/deletion. This is required to support the frame delimiting flags defined in section 4 of this specification. The flags are defined as the bit pattern 01111110. In order to guarantee uniqueness of this bit pattern, there may be no more than five consecutive ones in the bit stream (other than flags). This is accomplished using zero bit insertion (by the transmitter) and deletion (by the receiver). Specifically, the transmitter inserts a zero into the bit stream anytime it detects five consecutive ones (except for flags) regardless of the next bit value. Receivers then remove any zero from the bit stream that is preceded by five ones. The receiver must also detect the 01111110 bit pattern as a frame delimiting flag.

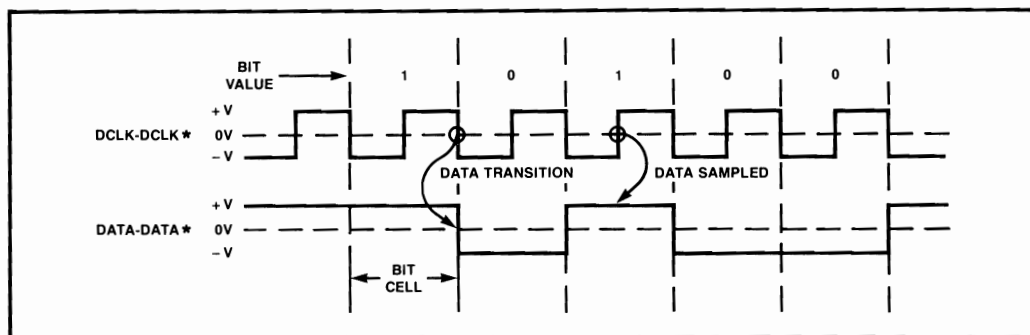


Figure 8. Synchronous Mode Data Encoding

3.1.3 SELF CLOCKED MODE

The self clocked mode of operation uses standard NRZI encoding with zero bit insertion/deletion. This encoding method combines serial data and serial clock information onto a single signal pair (DATA, DATA *). A logical zero is represented by a change in the polarity of the data signal pair from the preceding bit cell (i.e. a transition occurs on the bit cell boundary). A logical one is represented by no change in the polarity of the data signal pair from the preceding bit cell. These relationships are shown in figure 9. As in the synchronous mode, zero bit insertion/deletion is provided to guarantee uniqueness of the frame delimiting flag pattern.

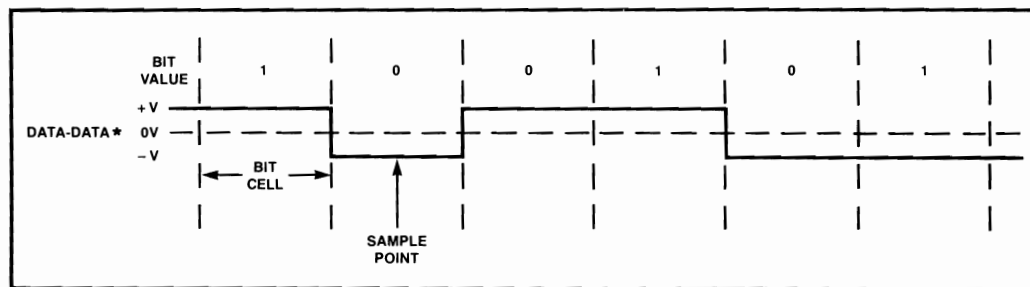


Figure 9. Self Clocked Mode Data Encoding

The self clocked mode of operation requires that receivers recover the serial clock from the data bit stream. The BITBUS interconnect is defined with the assumption that a digital phase locked loop (DPLL) with a $16 \times$ reference clock will be used for this purpose. For proper operation, the DPLL must be synchronized (i.e. be in phase with the transmitter clock) before, and remain synchronized during, the transmission of a frame. Initial synchronization of the receiver's DPLL is guaranteed by the transmission of a preframe sync (PFS) prior to the frame. The PFS consists of a minimum of eight zeros (i.e. transitions) that allow the DPLL to adjust (one reference clock at a time) until

synchronized. Once synchronized, the DPLL uses the transitions (i.e. zeros) within the bit stream to *make fine adjustments of ± 1 reference clock cycle*. Note that zero bit insertion/deletion guarantees

a transition at least every seven bit cells, allowing a reasonable tolerance on the reference clock.

3.2 DC SPECIFICATIONS

The DC specifications for the BITBUS interconnect are based on the RS485 electrical standard. This section defines the characteristics of a standard load, a transmitter, a receiver, the interconnect cable and the terminating and biasing resistors.

3.2.1 STANDARD LOAD SPECIFICATION

The standard unit load specification is used as a basis to define the load presented by a node to the BITBUS interconnect. (Figure 10 shows the I/V (current versus voltage) characteristics of the standard unit load. Note that this is 1.125 RS485 unit loads and it is specified over the input voltage range of +12 volts to -7 volts.

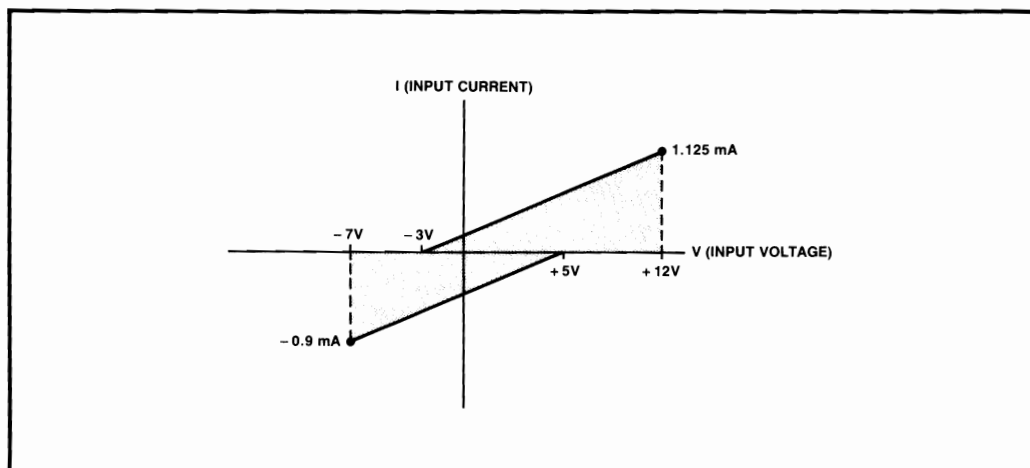


Figure 10. Standard Unit Data

The shaded region in figure 10 indicates the acceptable region for the I/V characteristic representing one standard unit load. Actual implementations may present a load which is a multiple of or fraction of this standard. The actual load specification of a connection is determined by plotting its I/V characteristics, and then scaling the I axis (i.e. multiply I axis by scaling factor) until the I/V characteristic is contained within the shaded region and touching the boundary. The factor used to scale the I axis is the number of standard unit loads.

The standard unit load is used to define the BITBUS interconnect load. The following sections specify how the standard unit load is applied to the various signal pairs, and how interface loads are computed for the synchronous and self clocked modes of operation.

3.2.1.1 Data And Data Clock Pair Specification

The load for the data pair (DATA, DATA *) and data clock pair (DCLK, DCLK *) consist of a disabled transmitter and a receiver as shown in figures 4, 6 and 7. The DC characteristics of these loads are specified as a multiple of the standard unit load shown in figure 10. The standard unit load specification has been defined such that most implementations with standard off the shelf components will be one standard unit load.

3.2.1.2 Transceiver Control Pair Specification

The load for the transceiver pair (RTS, RTS*) is not the same as the data pair. While data may be driven from the master to a slave, as well as from a slave back to the master, the transceiver signals are driven one direction only—from a slave back towards the master. This difference is shown in Figure 7.

A BITBUS segment may have up to 28 RTS/RTS* transmitters and one RTS/RTS* receiver. At any time, no more than one of these transmitter pairs is active. This active transmitter sees one receiver and up to 27 inactive transmitters.

The DC characteristics of the inactive transmitter loads are specified as a multiple of 1/9 the standard unit load shown in Figure 10. For example, if a transceiver node presents 2/9 standard unit loads, it is counted as 2 BITBUS interconnect loads. The DC characteristics of the active receiver load shall not exceed 25 standard unit loads, including the biasing resistors specified in section 3.2.5. With this definition, 27 inactive transmitter nodes and one active receiver node are equivalent to 28 standard unit loads ($27 \times 1/9 + 25 = 28$).

3.2.1.3 Synchronous Mode Load

The load presented to the BITBUS interconnect in synchronous mode is defined by determining the number of standard unit loads for each signal pair. The largest number is the specification. For example, if the data signal pair presents a load equal to the standard unit load, and the data clock signal pair presents a load equal to 2 standard unit loads, the interface would be specified as 2 loads.

3.2.1.4 Self Clocked Mode Load

The load presented to the BITBUS interconnect in the self clocked mode is also defined by determining the number of standard unit loads for each signal pair. For example, if the data signal pair presents a load equal to the standard unit load, and the transceiver control pair presents a load equal to 2/9 standard unit loads, the interface would be specified as 2 loads. If the transceiver control pair were not used (i.e. no repeater connections) the interface would be specified as 1 load.

3.2.2 TRANSMITTER SPECIFICATION

The standard BITBUS transmitter shall meet the requirements of an RS485 generator. It shall be capable of driving a 60 ohm termination (120 ohms at each end of the cable) and up to 32 RS485 unit loads (28 standard unit loads in Figure 10). For reference, the DC requirements are repeated in this specification. For the figures in this section all output voltages apply to both logic states and the symbol Δ in front of a voltage refers to the difference between that voltage in the two logic states.

3.2.2.1 Open Circuit Specification

The transmitter open circuit test configuration is shown in figure 11. All relevant test parameters are shown in the figure. This test is used to verify the output voltage characteristics when no load is applied.

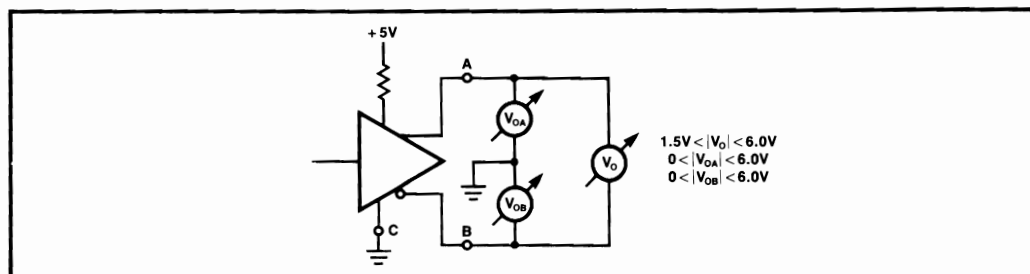


Figure 11. Transmitter Open Circuit Test Configuration

3.2.2.2 Test Termination Specification

The transmitter shall be able to drive the two test configurations shown in figures 12 and 13. All relevant test parameters are shown in the figures.

In figure 12 the transmitter drives a 54 ohm test load consisting of two 27 ohm resistors in series. This test termination is used to verify the output drive (V_O) capability and the signal pair offset voltage (V_{OS}) characteristics.

In figure 13 the transmitter drives a 60 ohm termination plus a load that simulates 32 RS485 unit loads through the RS485 common mode voltage range of +12 to -7 volts. This test termination is used to verify output drive capability under worst case command mode conditions.

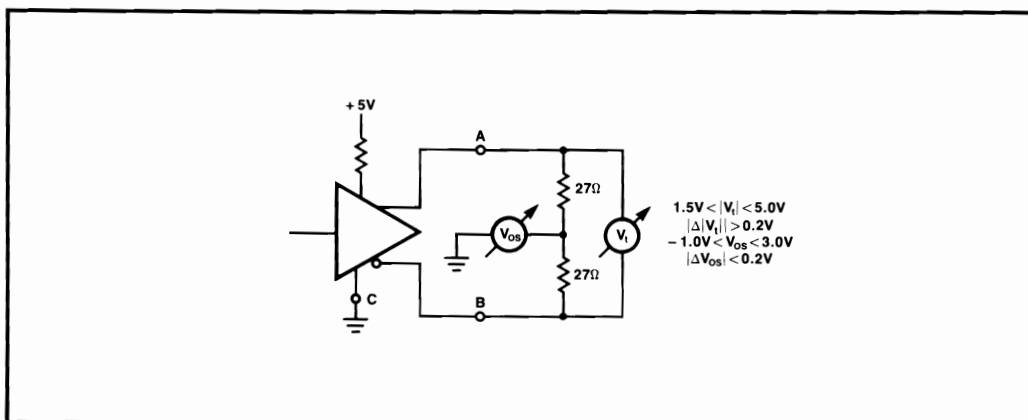


Figure 12. Transmitter Test Termination Configuration 1

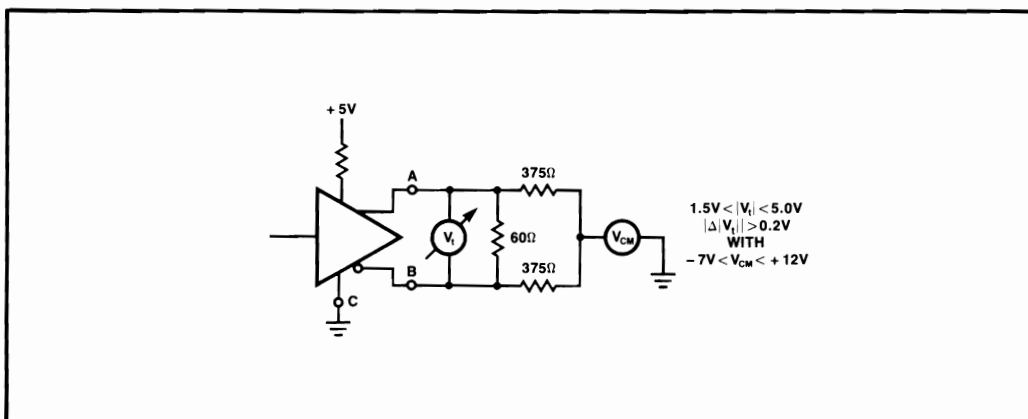


Figure 13. Transmitter Test Termination Configuration 2

3.2.2.3 Fault Conditions Specification

The transmitter shall be able to withstand, without permanent damage, the following two conditions:

- Outputs shorted together.
- Output(s) directly connected to a current limited voltage source within the +12 to -7 volt range.

Figure 14 shows the test configurations.

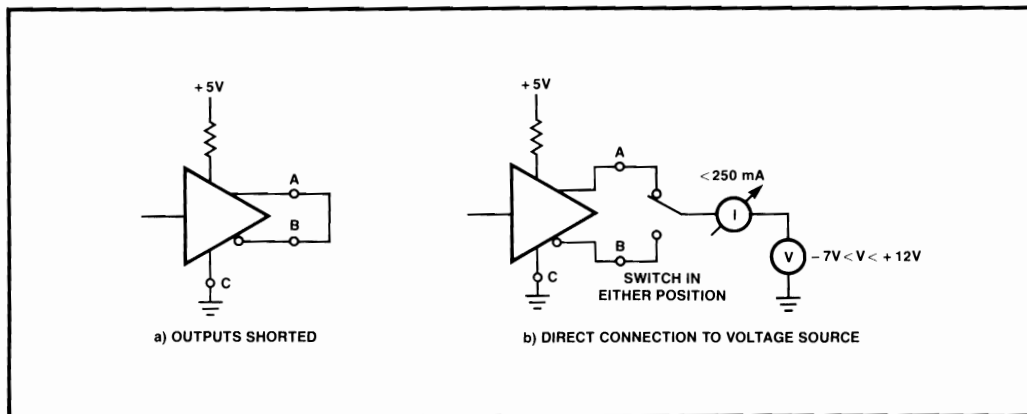


Figure 14. Transmitter Fault Conditions

3.2.3 RECEIVER SPECIFICATION

The standard BITBUS receiver shall meet the requirements of an RS485 receiver. Receiver load considerations are included in section 3.2.1. For reference, the input sensitivity and balance requirements are repeated in this section.

3.2.3.1 Sensitivity Specification

The receiver sensitivity specification is shown in figure 15. The allowable range of input voltages appearing at the receiver inputs, referenced to the receiver ground, shall be between +12 and -7 volts. Within this range a differential voltage of 200 millivolts or more shall be detectable as a valid logic state. The states are indicated in figure 15.

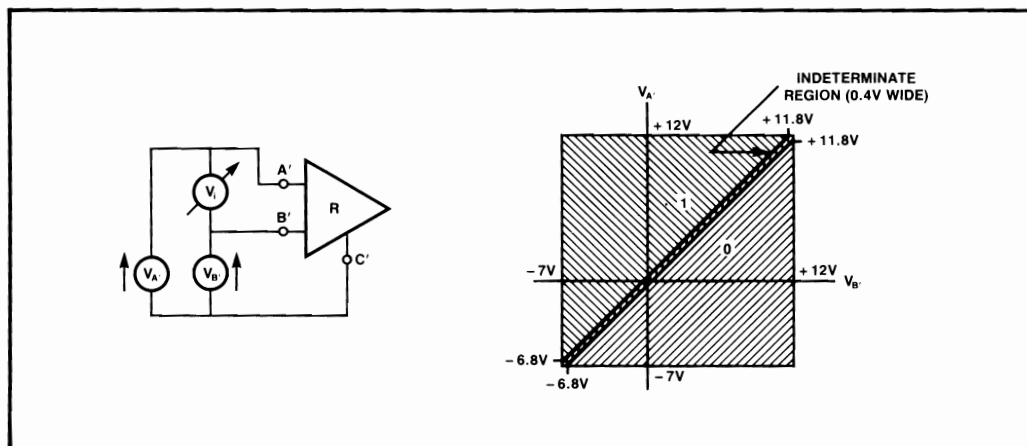


Figure 15. Receiver Sensitivity Specification

3.2.3.2 Balance Specification

The receiver balance specification is shown in figure 16. For this test the input resistor shown is 1500 ohms divided by the number of RS485 unit loads of the receiver. Within the common mode input range of +12 to -7 volts, a differential input of 400 millivolts or more shall be detectable as a valid logic state. The states are indicated in figure 16.

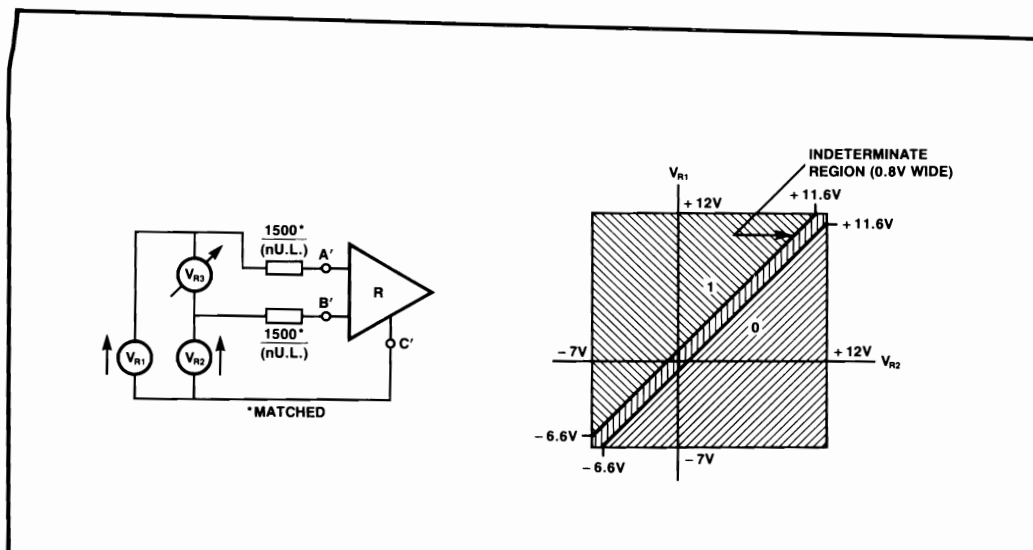


Figure 16. Receiver Balance Specification

3.2.4 CABLE CHARACTERISTICS

The BITBUS is defined to operate on a variety of cables including a low cost twisted pair (shielded or unshielded depending on application requirements) or backplane traces. Specific cable characteristics are not rigidly specified in order to allow flexibility for various implementations (i.e. only the interfaces are rigidly specified). When selecting a cable, the following aspects must be considered. Generally these parameters can be found in the cable data sheet.

- Characteristic Impedance
- Attenuation
- Resistance

3.2.4.1 Characteristic Impedance

Cables should be chosen with a characteristic impedance of 120 ohms or more to allow for properly matched terminations. Cables with lower characteristic impedance may be used in systems that guarantee low enough attenuation to prevent the reflections caused by the mismatch from switching the logic state. In severe environments, such as backplanes, greater mismatch can be tolerated as long as the maximum round trip propagation delay is less than one-half the signal rise time specified in section 3.3.1.1.

3.2.4.2 Attenuation

Cable must be chosen to support the desired bit rates over the desired distance. The specified BITBUS interconnect distances are based on the characteristics of readily available low cost twisted pair cables.

3.2.4.3 Resistance

Cable must be chosen with a low enough DC resistance to guarantee sufficient voltage across the termination to be detected by receivers. This parameter, the attenuation, and the magnitude of reflections, need to be considered together to determine the worst case noise margin at the receivers.

3.2.5 TERMINATION AND BIASING

The BITBUS interconnect requires the cable to be terminated for proper operation. There are two cases to be considered: the signal pair termination and the transceiver control pair biasing.

3.2.5.1 Termination

All BITBUS interconnect cables must be terminated at both ends for proper operation. The terminations shall be located at the extreme ends of the cable. The value of each termination shall be 120 ohms or greater and should be chosen to match the characteristic impedance of the cable as closely as possible.

3.2.5.2 Transceiver Control Biasing

In addition to terminations, the transceiver control pair shall have biasing resistors at its receiver only node. These resistors shall be 470 ohms $\pm 5\%$, one being connected to +5 volts $\pm 5\%$ and the other connected to ground as shown in figure 7. The receiver only node may be located anywhere on the cable segment.

3.3 AC SPECIFICATIONS

The AC specifications for the BITBUS interconnect requires definition of the signal line characteristics, transmitter enable timing, self clocked mode timing, and repeater timing.

3.3.1 SIGNAL LINE CHARACTERISTICS

The BITBUS signal lines must maintain a reasonable level of signal integrity to guarantee proper operation. Specifically, there needs to be bounded rise and fall times and reflection guidelines.

3.3.1.1 Rise And Fall Time Specification

The BITBUS signal lines shall have rise and fall times between 25 and 100ns as shown in figure 17. This measurement shall be made in the test configuration shown in figure 18.

In actual systems, the rise and fall times may be slower than the above specification as long as the following two conditions are met.

- The rise and fall time of any node shall meet the above specification (Figure 17) into the test load (Figure 18).
- The rise and fall times in the actual system shall not be more than 0.3 times the bit cell width, measured anywhere in the system.

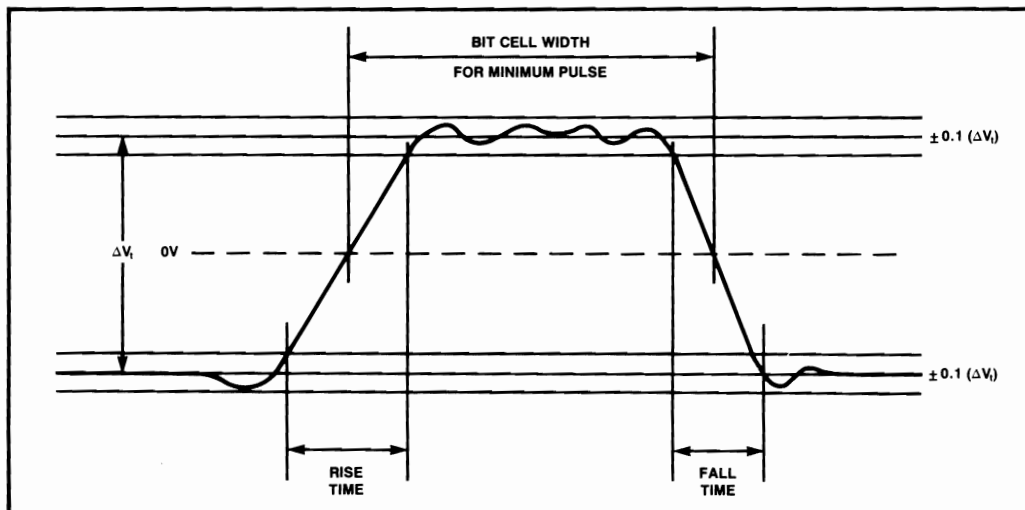


Figure 17. Rise and Fall Specification

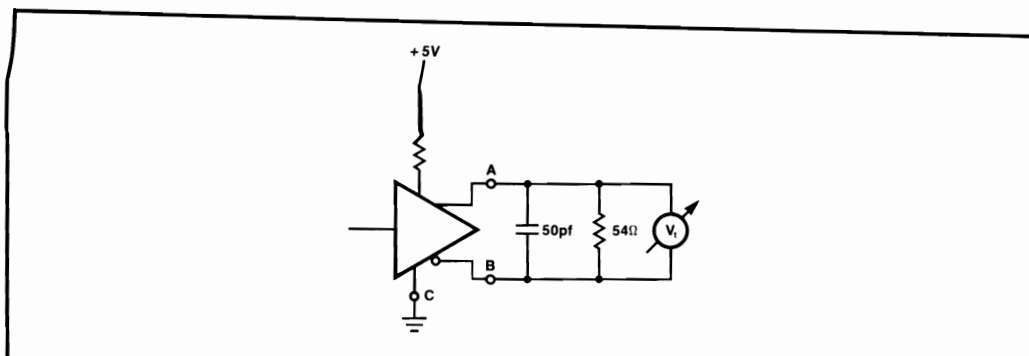


Figure 18. Test Configuration for Rise and Fall Time Measurement

3.3.1.2 Reflection Guidelines

Transitions on the BITBUS signal lines shall meet the requirements shown in figure 17 when driving the test load in figure 18. The transition from 10% to 90% of the final steady state value shall be monotonic. Ringing shall be limited to $\pm 10\%$ around the final steady state value.

RS485 is a robust standard, providing a significant DC noise margin. In practical systems where reflections may be unavoidable these noise margins shall be used to guarantee proper operation. Potential reflections (due to lumped capacitive loads, mismatched terminations, or excessively long stubs) should be considered together with the DC losses specified in section 3.2.4 when choosing cable. In all cases, the voltage at any node shall remain in the +12 volt to -7 volt common mode range except for pulses of less than 15 microseconds, at less than 1% duty cycle, which shall be bounded to ± 25 volts.

3.3.2 TRANSMITTER ENABLE TIMING

The BITBUS interconnect is designed to operate without transceiver control signals (repeaters excepted). In order to guarantee proper operation, this requires specifications for transmitter turn on and turn off in order to prevent a “dribbling” (enabled after end of frame) transmitter from corrupting the data of another transmitter. Figure 19 summarizes these parameters.

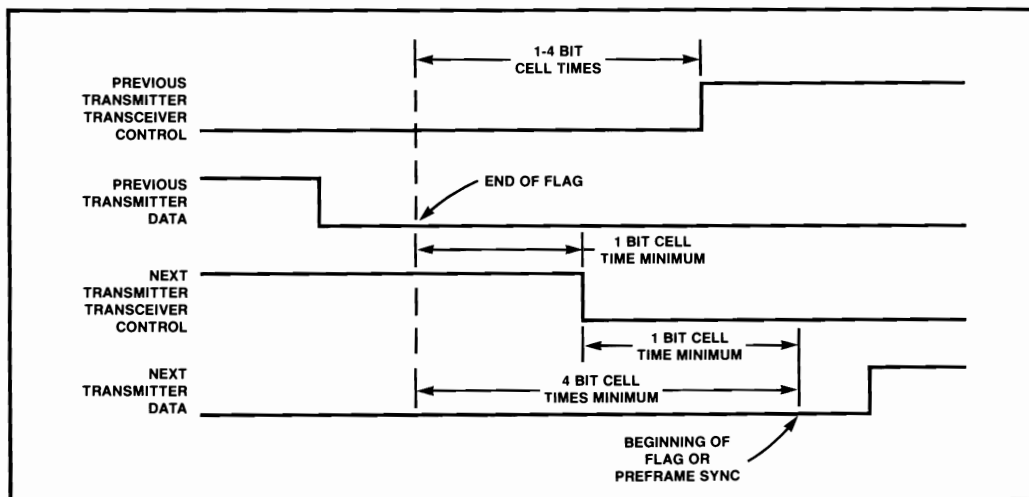


Figure 19. Transmitter Enable Timing

Transmitters shall be enabled a least one bit cell time prior to valid data transmission (opening flag in synchronous mode, preframe synch in self clocked mode). A transmitter shall not be enabled until at least one bit cell time after detecting the closing flag of a previous frame. Finally a transmitter shall guarantee that the first valid bit of data is not transmitted until any previous transmitter has been disabled.

Transmitters shall be disabled between one and four bit cell times after the end of the closing flag. Note that this specification leads to a potential three bit cell time contention period during turn around. This is acceptable based on the fault condition specification of section 3.2.2.3.

3.3.3 SYNCHRONOUS MODE TIMING

The synchronous mode of operation requires specification of the data clock signal pair timing and the data pair signal timing.

The data clock signal pair timing is shown in figure 20. All receivers shall be able to receive at speeds up to 2.4 Mbits/sec. Transmitters may transmit between 500 kbits/sec and 2.4 Mbits/sec.

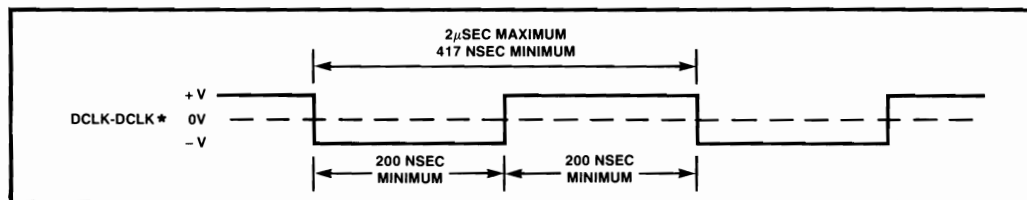


Figure 20. Data Clock Signal Pair Specification for Synchronous Mode

The data signal pair timing is shown in figure 21. The data signal pair is specified with respect to clock edges on the data clock signal pair. Data is changed on the “falling edge” and sampled on the “rising edge” of the data clock pair. These specifications assume balanced delays throughout the system. Specifically, the transmitters for the two signal pairs shall be in the same piece of silicon, the receivers for the two signal pairs shall be in the same piece of silicon and the conductors for the two signal pairs shall be of the same type, the same length and equally loaded.

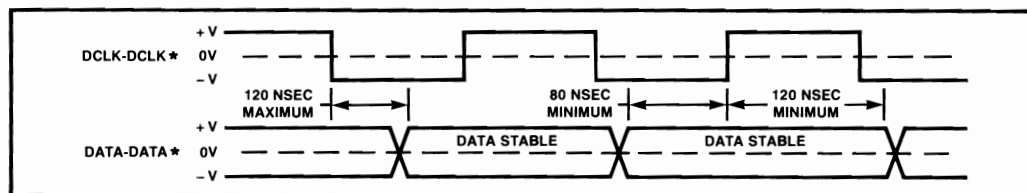


Figure 21. Data Signal Pair Specification for Synchronous Mode

3.3.4 SELF CLOCKED MODE TIMING

The self clocked mode of operation requires specification of the data signal pair and the transceiver control signal pair.

The data signal pair combines both data and clock information in the self clocked mode of operation. The clock information on the signal pair is the transmitter’s clock. The receiver uses a separate reference clock to recover the data. The tolerance of each of these clocks shall be $\pm 1\%$ for both standard bit rates (375 kbit/sec and 62.5 kbits/sec). The transmitter timing is shown in figure 22.

The transceiver control signal pair shall be driven whenever the transmitter is driven. These specifications are provided in section 3.3.2.

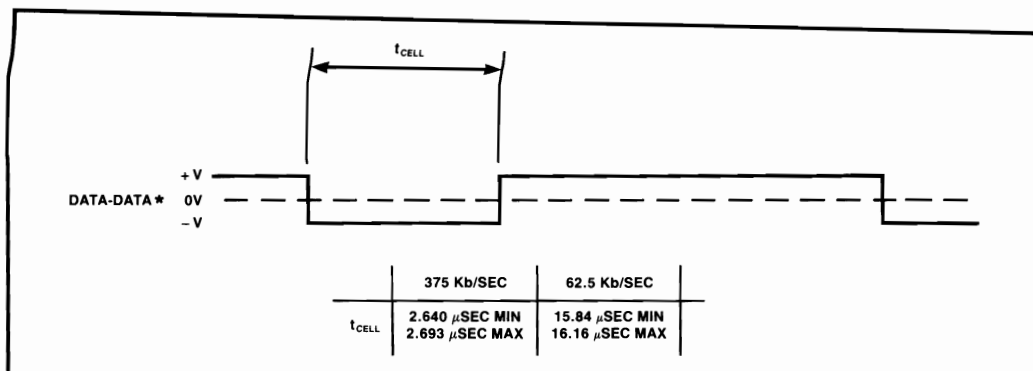


Figure 22. Data Signal Pair Specification for Self Clocked Mode

3.3.5 REPEATER TIMING

The self clocked mode of operation requires timing specifications for repeaters. Specifically, the skew between the two logic transitions and the skew between the data signal pair and the transceiver control pair need to be specified.

The worst case skew permitted through a repeater between the high to low transition time and the low to high transition time shall be less than ± 50 nsec as shown in figure 23. The worst case skew between the data signal pair and the transceiver control signal pair shall be less than ± 250 nsec as shown in figure 24.

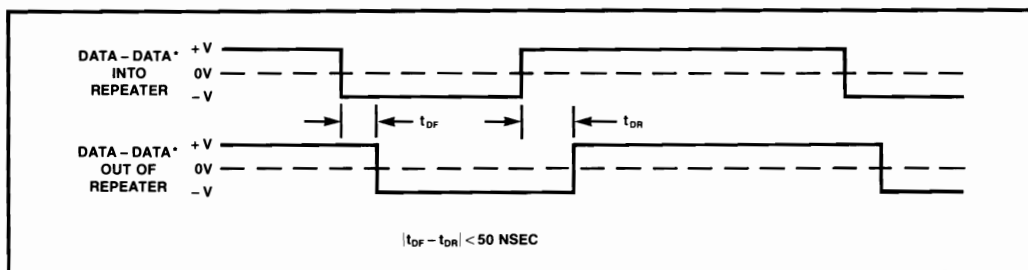


Figure 23. Repeater Skew on Transitions

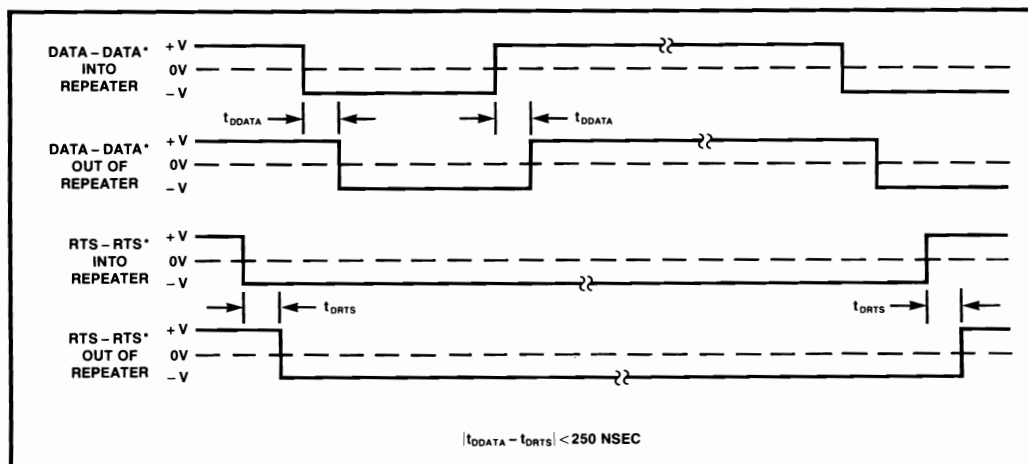


Figure 24. Repeater Skew Between Signal Pairs

The above specifications are for a standard repeater skew. Actual implementations of repeaters may be multiples of or a fraction of this standard. In all cases the specifications shown in table 1 shall apply for the maximum number of standard repeater skews between the master and any slave.

Table 1. Number of Repeaters

SPEED (kbit/sec)	MAX #OF STANDARD REPEATERS BETWEEN THE MASTER AND ANY SLAVE
375	2
62.5	10

4.0 DATA LINK PROTOCOL SPECIFICATION

The BITBUS data link protocol is a subset of the IBM Synchronous Data Link Control (SDLC) standard. SDLC is used as a basis for the BITBUS data link protocol because it is a highly reliable and proven protocol. The BITBUS interconnect does not (and is not intended to) maintain strict SDLC compatibility.

Based on SDLC, the BITBUS data link protocol connects a single master device to multiple slave devices in a multidrop (i.e. bus) topology. The data link protocol is specifically responsible for encapsulation of messages (messages are defined in section 5 of this specification) into frames and the control of frame transfer over the data link. To accomplish this, the specification defines the concept of system state, a general frame format, specific control fields and basic bus operations.

4.1 SYSTEM STATE

The ability of the BITBUS interconnect to connect nodes over a physically distributed domain requires some aspects of the system state to be formally defined. Being a hierarchical system, this state is defined from the point of view of the master device. This section provides an overview of the concept of system state. A detailed definition is provided in section 4.4.1 of this specification.

4.1.1 MASTER DEVICE STATE

The master device has full knowledge and control of its own state. Operation of a slave device does not require any information about the master device state. This implies that the master device state information is not transferred on the BITBUS interconnect and, therefore, is not a part of this specification.

4.1.2 SLAVE DEVICE STATE

The state of a slave device on the BITBUS interconnect needs to be known by the master device for proper operation. However, being physically separated, it is impossible for the slave device state to be precisely known at all times by the master device (e.g. in the event of a local reset, power down, etc.). In order to update the master device on a slave device state, it is necessary to define a state transfer mechanism over the BITBUS interconnect. The master device uses this mechanism to keep a record of the last known state for each slave device. This state is assumed for the next transfer and checked. If incorrect, the appropriate recovery action is taken. Slave device state consists of two parts: slave device mode and sequence count.

4.1.2.1 Slave Device Mode

A slave device is always in one of two modes: normal disconnect mode (NDM) or normal response mode (NRM). Below is a brief description of each mode. A formal state diagram is provided in section 4.4.1 of this specification.

A slave device enters NDM after a local reset or when it detects an irrecoverable protocol error. In this mode, a slave is awaiting a specific command from the master device to enter NRM. A slave

device may not exchange messages with the master device in this mode.

A slave device enters NRM only after receiving a specific command from the master device. Upon entering NRM, a slave device is "synchronized" with the master device, meaning that all sequence counts match (they are all initialized to 0). In this mode, a slave device may exchange messages with the master device as long as "synchronization" is maintained (i.e. no sequence count errors).

4.1.2.2 Sequence Counts

In NRM, sequence counts are used by the master device and each slave device to guarantee that frames are not lost or duplicated. Below is a brief description of how sequencing works. Further details are provided in section 4.3 of this specification.

Sequencing is performed by the master device and each slave device via two pairs of 3 bit sequence counts. Each slave keeps an N_r (number received) sequence count and an N_s (number sent) sequence count. The master device keeps a corresponding pair of counts for each slave it communicates with. A slave device is "synchronized" with the master device when the sequence counts are correct. The sequence counts at a slave device are considered part of the slave device state. The sequence counts at the master device are the master device's best knowledge of the slave device state.

The N_r sequence count indicates the sequence count of the next expected incoming message. The N_s sequence count indicates the sequence count of the next message awaiting acknowledgement (may or may not be outstanding). Each time a transfer occurs in NRM these numbers are included and verified resulting in three possible outcomes: correct sequence count, recoverable sequence error or irrecoverable sequence error. The irrecoverable sequence count error case requires the master device to resynchronize with the slave device by causing it to return to NDM, then re-enter NRM. The other cases allow the slave device to remain in NRM. Further details are provided later in this specification.

4.2 FRAME FORMAT

The data link protocol is responsible for creating frames to be transferred across the BITBUS interconnect. All frames use the frame format shown in figure 25. All fields are composed of one or more bytes (plus possible zero bit insertion), with the least significant bit (LSB) of each byte transmitted first. The only exception is the frame check sequence field, which transmits the most significant bit of each byte first.

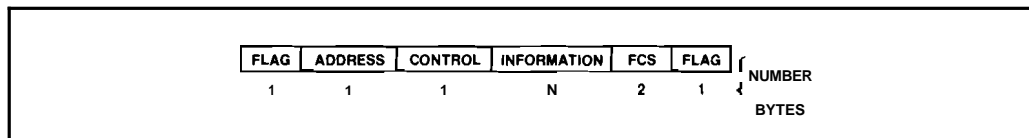


Figure 25. Standard Frame Format

4.2.1 FLAG FIELDS

The flag fields are used to delimit the frame. These fields (one at the beginning of the frame and one at the end of the frame) contain the unique bit pattern 01111110. The uniqueness of this bit pattern is guaranteed by using zero bit insertion/deletion in all other fields as described in section 3 of this specification. These fields are required in all frames.

4.2.2 ADDRESS FIELD

The address field contains the address of the slave device involved in the transfer. For a transmitting master device, this field identifies the destination slave device. For a transmitting slave device, this

field identifies the source slave device to the master device. If this field does not match the slave device address, the frame is ignored. The address field is eight bits long and may contain values from 0 to 255. Values 0 and 251-255 are reserved by Intel. All others may be used without restriction. This field is required in all frames.

4.2.3 CONTROL FIELD

The control field is used for command and status exchange between the master device and slave devices. This field is eight bits long and is used for three classes of operations: synchronization, supervision and message transfer. Below is an overview of these operations. Details are provided in section 4.3 of this specification. This field is required in all frames.

4.2.3.1 Synchronization

The transfer of sequenced messages between the master device and a slave device requires that the slave device be properly synchronized to the master device. This synchronization process is performed using unnumbered frames (i.e. frames with unnumbered control fields). As the name implies, unnumbered frames do not use the sequencing feature.

4.2.3.2 Supervision

After the master device is synchronized with a slave device, it is often necessary to exchange status information in the absence of messages. This is done with supervisory frames (i.e. frames with supervisory control fields). These frames are used by the master device to poll a slave device and by a slave device to acknowledge receipt of a valid frame (i.e. address match and no CRC error) from the master device.

4.2.3.3 Information

Information frames (i.e. with information control fields) are used by the master device or a slave device to transfer messages (messages are defined in section 5 of this specification). These frames are only used after synchronization, as are supervisory frames. In addition to a message and its sequence count, information frames carry the same status information as supervisory frames. In fact, information frames may be considered a **superset** of supervisory frames.

4.2.4 INFORMATION FIELD

The information field is used to carry the **BITBUS** message. Details of this field are provided in section 5 of this specification. This field is required for information frames and is not used for supervisory or unnumbered frames.

4.2.5 FRAME CHECK SEQUENCE FIELD

The frame check sequence (FCS) field provides the lowest level of error detection on the **BITBUS** interconnect. This field contains a 16 bit cyclic redundancy check (CRC). The transmitting node generates and sends this field, while the receiving node checks it for correctness. A receiving node ignores an incoming frame if the CRC is incorrect. The CRC is generated by the standard **CRC-CCITT** polynomial $X^{16} + X^{12} + X^5 + 1$. This field is required in all frames. Unlike other fields, the most significant bit of each byte is transmitted first, and the most significant byte is also transmitted first.

4.3 CONTROL FIELD DEFINITION

Operations on the **BITBUS** interconnect are performed using three types of control fields: unnumbered, supervisory and information. This section specifies these fields in detail and explains their usage.

For reference to SDLC, this specification uses a subset of the defined control fields with the **poll/final** bit always set. This implies that frames sent by the master device always expect a response from

the addressed slave device (poll bit set) and that frames sent by slave devices always return control of the link (final bit set) to the master device.

4.3.1 UNNUMBERED FRAMES

Unnumbered frames are used on the BITBUS interconnect for synchronizing slave devices with the master device. This section specifies the control field format for these frames and defines the unnumbered operations (commands and responses) supported on the BITBUS interconnect.

4.3.1.1 Unnumbered Control Field Format

The control field format for unnumbered frames is shown in figure 26. This field may specify one of several unnumbered operations.

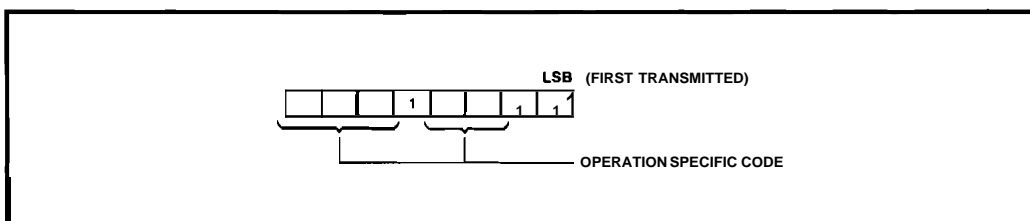


Figure 26. Unnumbered Control Field Format

4.2.1.2 Unnumbered Operations

The BITBUS interconnect supports two unnumbered commands (SNRM and DISC) and two unnumbered responses (UA and FRMR). The control fields for these operations are summarized in table 2.

Table 2. Unnumbered Control Fields

OPERATION	COMMAND	RESPONSE	CONTROL VALUE FIELD
FRMR			97H

The frame reject (FRMR) command is sent to the master device by a slave device that detects an invalid control field in an otherwise valid frame. It is also used by a slave to respond to any unnumbered frame while in NRM, any supervisory or information frame while in NDM, any unsupported control field, or an irrecoverable sequence count error. Upon receiving this command, the master device initiates resynchronization with the slave device.

The unnumbered acknowledge (UA) response is used by a slave device to acknowledge receipt of a valid unnumbered command while in NDM.

The disconnect (DISC) command is sent by the master device to a slave device to initiate **resynchronization**. This command causes the slave device to go to, or stay in, NDM. It is used by the master device when it detects the need to resynchronize (e.g. after reset, irrecoverable sequence error, etc. or in response to an FRMR from a slave device). When the master device receives a UA in response to a DISC it knows that the addressed slave device is in NDM.

The set normal response mode (SNRM) command is sent by the master device to synchronize a slave device. If a slave device is in NDM, this command causes it to enter NRM, allowing it to exchange messages with the master device. If a slave device is already in NRM, this command is invalid, causing the slave to reply with FRMR and enter NDM.

4.3.2 SUPERVISORY FRAMES

Supervisory frames are used on the BITBUS interconnect for passing status between the master device and a slave device while in NRM. These frames are used by a master device to poll slave devices and by slave devices to acknowledge receipt of a valid frame from the master device. This section specifies the control field format for these frames and defines the specific control fields that are supported.

4.3.2.1 Supervisory Control Field Format

The control field format for supervisory frames is shown in Figure 27. This field is used to specify the supervisory operation (RR or RNR) as well as to acknowledge receipt of a previous message via the N_r sequence count.

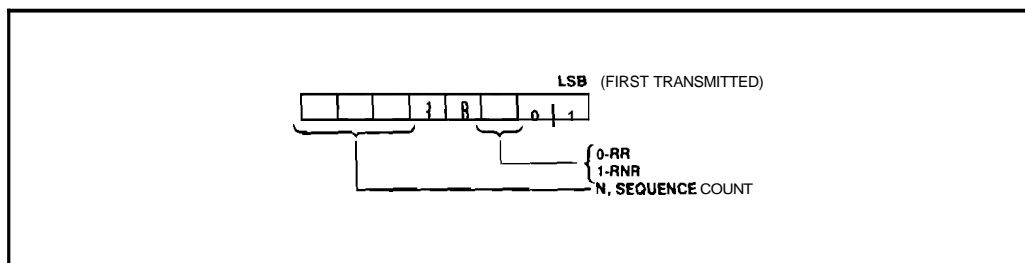


Figure 27. Supervisory Control Field Format

4.3.2.2 Supervisory Operation

The BITBUS interconnect supports two supervisory operations: receiver ready (RR) and receiver not ready (RNR).

The master device uses the RR supervisory frame to poll a slave device for an information frame. Slave devices use the RR supervisory frame to acknowledge valid reception (i.e. address match, valid CRC and available buffer space) of a previous supervisory or information frame when a message is not available (If a message is available an information frame is sent unless the incoming frame was a RNR supervisory frame).

The RNR supervisory frame is used to indicate that a buffer is presently unavailable to receive an otherwise valid frame. The master device uses the RNR supervisory frame to poll a slave device for data link status only. A slave device shall not respond to a RNR supervisory frame with an information frame. A slave device uses the RNR supervisory frame to indicate that the last frame was re-recognized, however there was no buffer available to store it. In this case the master device shall retransmit the frame.

Both RR and RNR frames contain an N_r sequence count. This sequence count acknowledges receipt of N_r-1 frames (i.e. N_r equals the next expected incoming frame). The receiver of a supervisory frame shall always compare this value to its N_s sequence count for correctness.

In the case that all messages have been acknowledged, the incoming N_r sequence count should equal the N_s sequence count. If the previous frame sent by the receiving node was an information frame, then the incoming N_r sequence count should equal the N_s sequence count plus 1. If this case is true, the previous frame is acknowledged meaning the N_s sequence can be incremented and the message buffer released. If after an information frame the incoming N_r sequence count is equal to the N_s sequence count, the information frame was not received and shall be retransmitted. Any other N_r sequence count value is detected as an irrecoverable sequence error requiring the master device to resynchronize with the slave device.

4.3.3 INFORMATION FRAMES

Information frames are used on the *BITBUS* interconnect for message transfer. These are the only

frames that contain an information (**I**) field. The content of the **I** field is the *BITBUS* message as defined in section 5 of this specification. This section specifies the control field format for these frames and defines the operation of the sequence counts.

4.3.3.1 Information Control Field Format

The control field format for information frames is shown in Figure 28. The control field contains an N_r sequence count to acknowledge valid reception of $N_r - 1$ frames and an N_s sequence count corresponding to the attached information field (i.e. message).

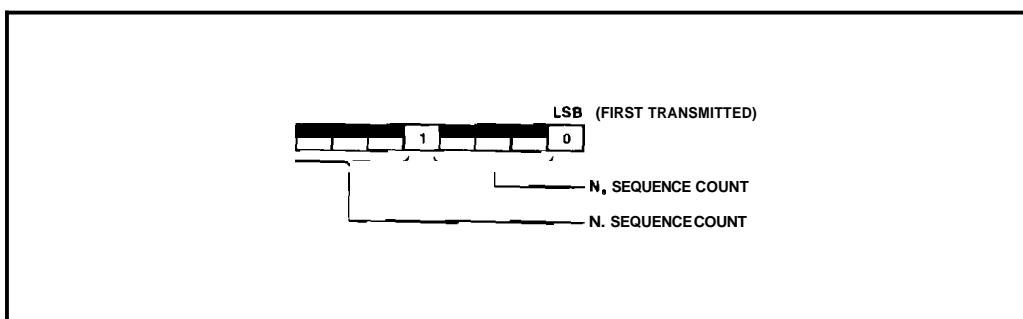


Figure 28. Information Control Field Format

4.3.3.2 Information Operation

Information frames are a **superset** of supervisory frames. They carry the same acknowledgement status as supervisory frames via the N_r sequence count. This count shall always be compared to the receiver's N_s sequence count as it is for supervisory frames.

In addition to the N_r sequence count, information frames carry a message in the information field and a corresponding N_s sequence count. Upon receiving an information frame, a receiver shall compare the incoming N_s count with its N_r count. If the counts match, the message shall be passed on. If the counts do not match, error recovery action shall proceed as follows. If the incoming sequence count N_s equals the receiver's sequence count $N_r - 1$, the receiving node may acknowledge receipt of the frame, then discard it (this case occurs when an information frame is received and its acknowledgement is corrupted, making the second information frame a duplicate). If the incoming sequence count N_s does not equal the receiver's sequence count N_r or $N_r - 1$, then an irrecoverable sequence error is detected. This case requires the master device to resynchronize with the slave device before further message transfers can occur.

4.4 DATA LINK OPERATION

This section summarizes the data link operation on the *BITBUS* interconnect. This summary includes a precise definition of slave state, slave state transition, and sequenced operations in NRM. In addition, sample synchronization sequences, sample transfer sequences, and a summary of error conditions and the corresponding recovery actions are presented.

4.4.1 SLAVE STATE DESCRIPTION

The state transition diagram for a slave device is shown in Figure 29. As previously discussed, the state diagram has only two basic states: normal response mode (NRM) and normal disconnect mode (NDM). Table 3 lists the responses to all possible incoming frames in each of the states. Note that this

table assumes that the slave device has recognized a valid incoming frame. That is, the address field matched the node address and the CRC field was correct.

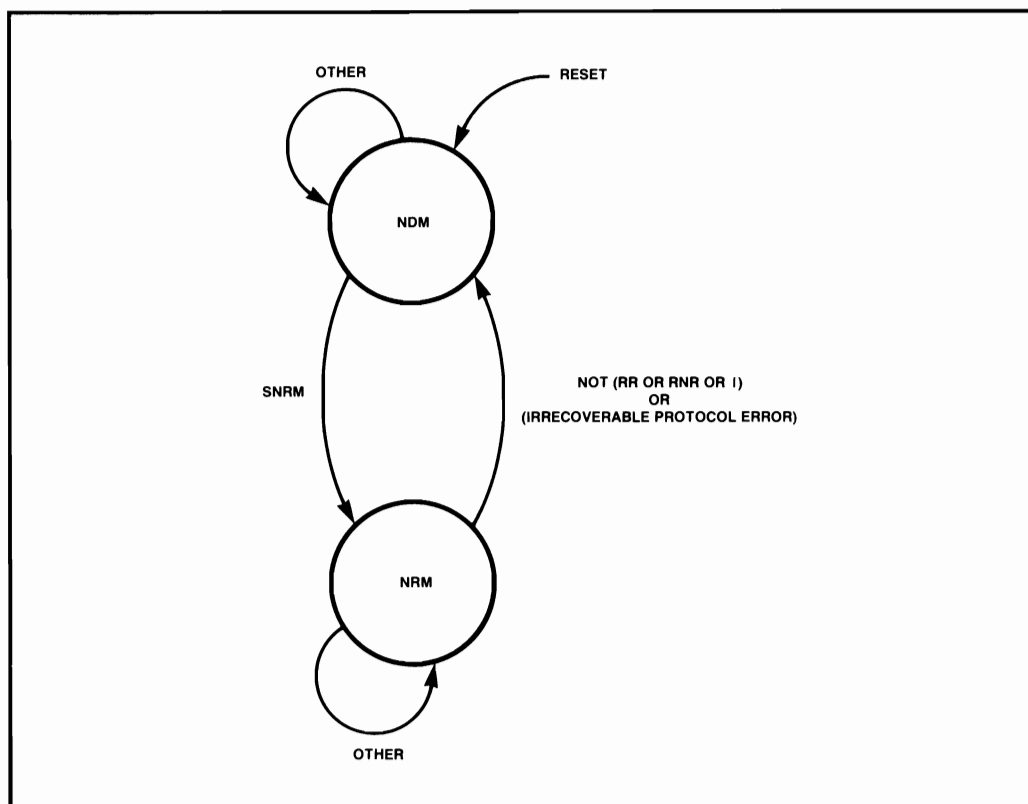


Figure 29. Slave Device State Diagram

Table 3. Slave Device State Transitions and Responses

SLAVE STATE	INCOMING FRAME	NEXT STATE	RESPONSE
NDM	DISC	NDM	UA
NDM	SNRM	NRM	UA
NDM	Other	NDM	FRMR
NRM	INFORMATION	NRM	RR,RNR,I
NRM	RR	NRM	RR,RNR,I
NRM	RNR	NRM	RR,RNR
NRM	INFORMATION,RR,RNR with irrocoverable sequence error	NDM	FRMR
NRM	Other	NDM	FRMR

When in NRM, the slave device state also includes its N_r and N_s sequence counts. For simplicity, these states are not shown, but instead, a flow chart is provided in figure 30. This flow chart, along with table 3, completely specifies the slave device interface.

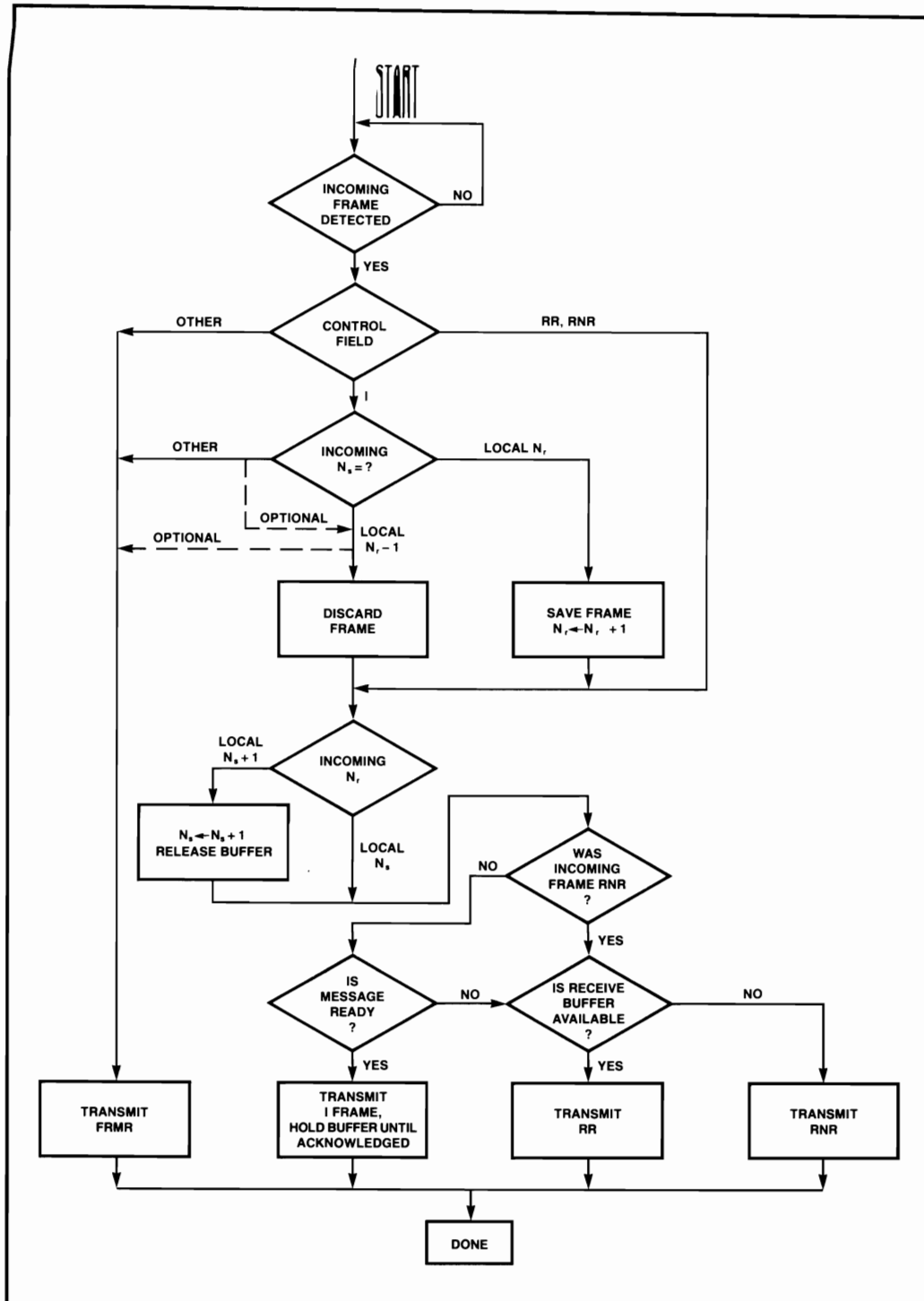


Figure 30. Slave Response to Incoming Frame in NRM

4.4.2 SYNCHRONIZATION SEQUENCE

The synchronization sequence on the BITBUS interconnect is used to establish a known state at the slave device (NRM with sequence counts equal to zero), in order to allow sequenced message transfer. Synchronization is required any time the master device is reset or any time the master device or slave device detects an irrecoverable protocol error. Synchronization is strictly a data link protocol feature and, therefore, does not appear at any higher levels of this specification. Figure 31 shows a typical synchronization sequence.

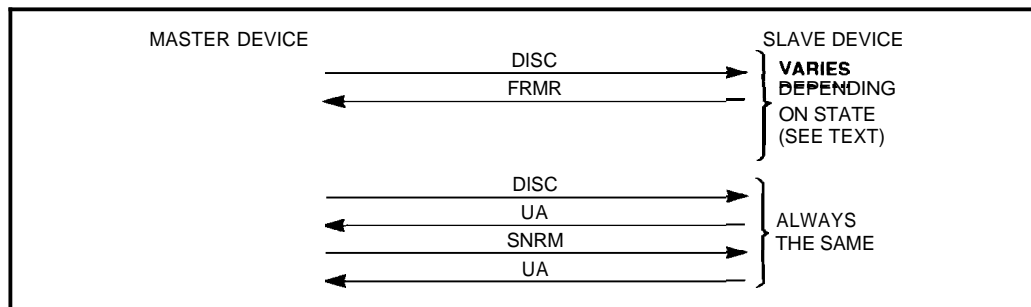


Figure 31. Typical Initialization Sequence

The master device may initiate the synchronization sequence by sending a DISC. The response from the slave device to a disconnect is a FRMR if it is in NRM or a UA if it is in NDM. The master device shall continue sending DISC frames until a UA response is returned verifying that the slave is in NDM. The master device then completes the sequence by sending an SNRM. Upon receiving the SNRM while in NDM, a slave device enters NRM and responds with a UA. Once the master device receives a UA to an SNRM, it knows that the slave device is synchronized.

The slave device may initiate the synchronization sequence by responding to an incoming frame with a FRMR frame. The master device responds to a FRMR with a DISC and proceeds through the sequence described above.

4.4.3 TRANSFER SEQUENCES

This section provides several transfer sequence examples. These examples serve as a basis for the message protocol defined in section 5 of this specification. Three examples are provided in figure 32: message transfer from a master device to a slave device, message transfer from a slave device to a master device and piggybacking of messages on acknowledges or polls. For these examples, it is assumed that the slave device is synchronized and that incoming frames are valid. Errors are discussed further in section 4.4.4 of this specification.

In the first example, the master device sends a message to a slave device. The information frame that carries the message also carries the sequence counts $N_t = 0$ and $N_r = 0$. The slave device verifies that the incoming N_r equals its N_t and that the incoming N_t equals its N_r . The slave then increments N_r to indicate it received the information frame and returns an RR frame with sequence count $N_r = 1$ to acknowledge the frame. Upon receiving the N_r sequence count, the master device sees that its information frame has been received, increments its N_t sequence count for the next message and releases the transmit buffer.

In the second example, the master device is polling a slave device with an RR frame for a message. Upon receiving the RR frame, the slave device verifies the incoming N_r sequence count and responds with an information frame containing the message and sequence counts $N_t = 0$ and $N_r = 1$.

Upon receiving the information frame, the master device verifies the incoming N_r and N_s sequence counts. It then increments its N_r to indicate that it has received the information frame. On the next transmission to the slave device, the master device sends its sequence count $N_r = 1$ to acknowledge receipt of the information frame. Upon detecting this, the slave device increments its N_s sequence count and releases the transmit buffer.

In the third example, the link efficiency is increased by piggybacking acknowledges and polls onto information frames. At both master device and slave device, an incoming frame carries an N_r sequence count which causes the N_s sequence count to be incremented and the last transmit buffer released. In addition, the N_s sequence count causes the N_r sequence count to be incremented. These results are then returned on the next information frame and the process is repeated.

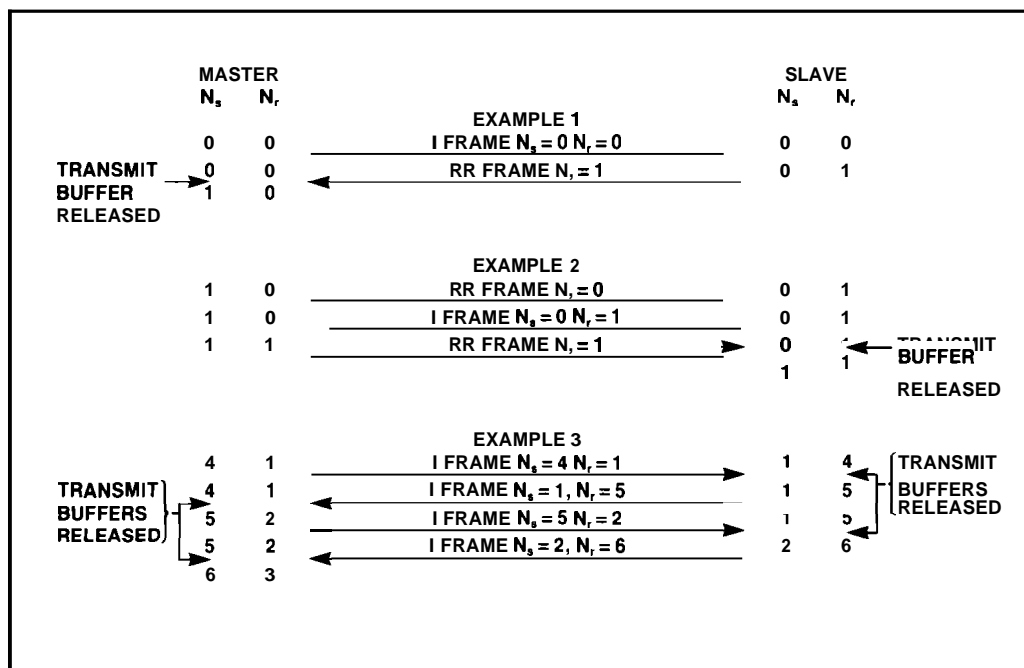


Figure 32. Transfer Sequence Examples

4.4.4 DATA LINK ERROR SUMMARY

The following section summarizes the possible data link error conditions and specifies the corresponding action. Specific error conditions are timed out, invalid control field, and sequence count.

4.4.4.1 Time Out

A time out error occurs when the master device does not receive a response to a transmitted frame within 10 milliseconds. This time is measured from the time transmission is completed until the response is received and the CRC verified.

Time out errors can occur under the following conditions: the addressed slave does not exist, the address slave discards the frame due to a CRC error, or a CRC error is detected by the master device in a response. In all cases the recovery action taken by the master device is to retransmit the frame a second time. If two consecutive errors are detected an irrecoverable protocol error is returned to the user.

4.4.4.2 Invalid Control Field

An invalid control field detected by either the master device or a slave device is considered an irrecoverable protocol error. Detection of this error condition at a slave device causes the slave device to enter NDM and respond with a FRMR. Detection at the master device causes it to assume that the slave is in NDM, requiring resynchronization.

4.4.4.3 Sequence Count

Sequence count errors are recoverable in some cases as indicated in figure 30. Cases that are not recoverable are considered irrecoverable protocol errors. Detection of an irrecoverable condition at a slave device causes it to enter NDM and respond with a FRMR. Detection at the master device causes it to assume that the slave is in NDM, requiring resynchronization.

5.0 MESSAGE PROTOCOL SPECIFICATION

The BITBUS message protocol is designed to provide a task to task message interface between a master node and multiple slave nodes using an order/reply structure. That is, the master node issues orders to slave nodes which respond with replies. This structure is built on top of the data link protocol using information frames for the message transfer. This section specifies the order/reply structure (including error handling) and the standard message format.

5.1 ORDER/REPLY STRUCTURE

The BITBUS message protocol is based on an order/reply structure in a multitasking environment. An order is defined as a message that is sent by a task on the master node, over the BITBUS interconnect, to a task on a slave node. A reply is defined as a message that is sent by a task on a slave node, over the BITBUS interconnect, to a task on the master node, in response to an order from that master node task. Master node tasks may exist on either the master device or master device extension (collectively referred to as the master node). Likewise, slave node tasks may exist on either the slave device or slave device extension (collectively referred to as the slave node). Note that orders and replies may have a more general definition when used between tasks on a given node. This special case is not part of this specification.

Every order on the BITBUS interconnect requires a corresponding reply; however, replies need not be returned in the same sequence that orders are issued. For example, if the master print node issues orders A,B,C to a slave node, the responses may be returned as A,B,C or B,C,A or C,A,B, etc. Based on this requirement, a master device need only poll a slave node when one or more orders are outstanding. This condition exists when the sequence counts N_r and N_s for a given slave node are not equal. When the sequence counts N_r and N_s are equal, the slave node has no more replies and need not be polled. This algorithm limits unnecessary polling and increases system performance.

The data link protocol provides sequence counts that are 3 bits long. This limits the number of outstanding messages to any given slave node to seven. The slave node is responsible for enforcing this limit by not providing a receive buffer for the eighth message.

The BITBUS message protocol provides sufficient control to route orders from the master node to a slave device or slave device extension, and route replies from a slave node to a master device or master device extension. This capability allows local messages within a node to co-exist in a common gateway implementation. Furthermore, multiple level hierarchies (as shown in figure 1) are easily supported using the same master device software. The details of these implementations are not part of this specification.

Several sources of error exist in the message protocol. Specific examples are irrecoverable protocol errors or missing tasks. In all cases, the BITBUS message protocol detects the errors and handles

them in the same way. If an error occurs while delivering an order, the complete order is immediately converted to a reply and returned to the originating task. The reply is identical to the order **except** for the error code in the response field. If an error is detected on a reply, it is simply discarded (note that this is a very rare case, except for catastrophic error, since the reply retraces the path of the order). In either case, there is a small possibility that a reply is not returned. The task originating the order must account for this by setting a time out period for recovery. Note that this time out is not related to the data link time out specified in section 4 of this specification. Its value is determined by the application.

5.2 MESSAGE FORMAT

The standard message format on the **BITBUS** interconnect is shown in figure 33. This message structure is sent across the **BITBUS** interconnect in the information field of information frames. All fields shown, except the data field, are required in all messages.

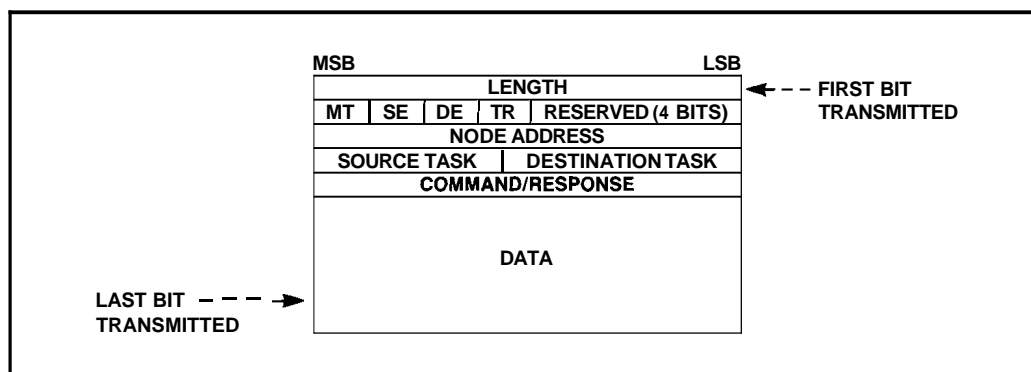


Figure 33. Message Format

5.2.1 LENGTH

The length field specifies the total message length. This eight bit field may contain values between 7 and 255. The value in this field equals the number of bytes in the data field plus 7. This value allows implementation to easily add two bytes for local message manipulation such as buffer control or queuing. All implementations shall support a data field of up to 13 bytes, corresponding to a length field equal to 20.

5.2.2 MESSAGE TYPE (MT)

The message type field is used to specify whether the message is an order or a reply. The master node always sends orders and, therefore, shall clear this bit to 0. A slave node always sends replies and, therefore, shall set this bit to 1.

5.2.3 SOURCE EXTENSION (SE)

The source extension field indicates whether the source of an order, or the destination of a reply, is the master device or the master device extension. This bit is set to 1 to indicate the master device extension and cleared to 0 to indicate the master device. This bit is unchanged between an order and its corresponding reply.

5.2.4 DESTINATION EXTENSION (DE)

The destination extension field indicates whether the destination of an order, or source of a reply, is a slave device or a slave device extension. This bit is set to 1 to indicate a slave device extension

and cleared to 0 to indicate a slave device. This bit is unchanged between an order and its corresponding reply.

5.2.5 TRACK (TR)

The track field is used to provide message control at a master or slave device which may be required by some implementations. This bit is cleared to 0 when sending a message. It is set to 1 upon receiving a message from the BITBUS interconnect.

5.2.6 RESERVED

These four bits are reserved for possible future enhancements. They shall be cleared to zero when sending a message and their value is not guaranteed upon receiving a message.

5.2.7 NODE ADDRESS

The node address field specifies the destination node for orders and the source node for replies (i.e. it is unchanged between order and reply). This eight bit field contains the same value as the address field in the data link frame. Valid entries are 1 through 250. The values 0 and 251-255 are reserved by Intel.

5.2.8 SOURCE TASK

The source task field identifies the task that has generated an order or is to receive a reply. This four bit field allows up to sixteen tasks to generate orders from the master device and the master device extension. Specific implementations may support as few as one or as many as sixteen tasks.

5.2.9 DESTINATION TASK

The destination task field identifies the task that is to receive an order or has generated a reply. This four bit field allows up to sixteen tasks to receive orders at each slave device and slave device extension. Specific implementations may support as few as one or as many as sixteen tasks.

5.2.10 COMMAND/RESPONSE

The command/response field is used by both user tasks and the message protocol. Under normal conditions, this field is used only by the user tasks. The message protocol only uses this field for reporting errors. Table 4 indicates the usage of this field. Note that this table only applies to responses. Commands are always defined by the user task.

Table 4. Message Protocol Responses

CONDITION	ERROR
No error (set by user)	00H
User defined	01H-7FH
No destination task	80H
Protocol Error	91H
No destination device	93H
Reserved by Intel	81H-90H, 92H, 94H-0FFH

5.2.11 DATA

The data field is defined by the contents of the command field. Minimum support requires this field to be capable of handling 13 bytes. Implementation may extend it to as much as 248 bytes provided that the longer messages are not sent to nodes that cannot support them. This field is the only optional field in the message.

6.0 REMOTE ACCESS AND CONTROL SPECIFICATION

The remote access and control (RAC) interface for the BITBUS interconnect defines a set of high level commands and responses to perform general purpose operations at a slave node. The remote access interface allows I/O read and write capabilities as well as memory download and upload. The remote control interface allows control and monitoring of tasks at a slave node. The RAC interface is built on top of the message protocol defined in section 5 of this specification. In fact, RAC is simply a special task defined at slave nodes. This section specifies the commands, responses and the general RAC message format.

6.1 MESSAGE FORMAT FOR RAC

The RAC interface is built on top of the standard message protocol. The message format follows the standard format in figure 33 with the following fields further defined: destination task, command/response, and data.

6.1.1 DESTINATION TASK

The BITBUS interconnect, by convention, defines task 0 as the RAC task. This makes its presence easily detectable and allows it to be addressed in the same way for all implementations. If the RAC task is not supported at a slave node, task 0 may be used for another function.

6.1.2 COMMAND/RESPONSE

The command/response field is used to identify the RAC command in an order message and the RAC response in reply messages. The various commands and responses supported by the RAC interface are specified in sections 6.2 and 6.3, respectively.

6.1.3 DATA

The contents of the data field are determined by the RAC command. The length field is used to identify the exact number of data bytes. The minimum BITBUS interconnect support provides up to 13 bytes in this field. Details of this field are provided in sections 6.2 and 6.3 of this specification.

6.2 RAC COMMANDS

Table 5 lists the RAC commands supported on the BITBUS interconnect. Commands 00H through 0BFH are general purpose commands controlled by Intel. At this time, 15 are specifically defined, and the rest are reserved. Commands 000H through 0FFH are available to the user for custom RAC commands. These values will not be standardized. Specific implementations of RAC may support all or any subset of the defined commands. The only specific requirements are that the implementation return an error response for unsupported commands and that all supported general purpose commands conform to this specification.

6.2.1 CREATE TASK

The create task command is used to initialize and begin execution of a task at a slave node. This command assumes that the task to be created is already in the slave node memory. The command is simply the mechanism used to inform the operating system at the slave node that a given task in memory is to become active.

The data field in a create task order contains an address pointer (length field equal to 7 plus pointer length) to the task descriptor in the slave node's memory. The length of the pointer is determined by the implementation requirements at the slave node. This pointer is sent most significant byte first. The actual task descriptor may vary between implementations and, therefore, is not part of this specification.

Table 5. RAC Commands

COMMAND	ACCESS	CONTROL	VALUE
Reset Slave		✓	00H
Create Task		✓	01H
Delete Task		✓	02H
Get Function ID		✓	03H
RAC Protect		✓	04H
Read I/O	✓		05H
Write I/O	✓		06H
Update I/O	✓		07H
Upload Memory	✓		08H
Download Memory	✓		09H
OR I/O	✓		0AH
AND I/O	✓		0BH
XOR I/O	✓		0CH
Status Read	✓		0DH
Status Write	✓		0EH
Reserved			0FH-0BFH
User Defined			0COH-OFFH

After successful completion of a create task command, a reply message is returned to the originating task on the master node with 0CH in the response field and the data field unchanged (i.e. address pointer is returned).

6.2.2 DELETE TASK

The delete task command performs the inverse function to the create task command. It is used to terminate execution of a task at a slave node. **This** allows the task number associated with the deleted task to be reused (i.e. with a create task command).

The data field in a delete task order contains a single byte (length field equal to 8) identifying the task number to be deleted. This field may contain values between 0 and 15; however, beware of deleting task 0 as it is the RAC task.

After successful completion of a delete task command, a reply message is returned to the originating task on the master node with 0CH in the response field and the data field unchanged (i.e. eight bit task number is returned).

6.2.3 GET FUNCTION IDs

Function IDs are logical IDs used to identify a given task by its function rather than its physical address. The concept of function ID allows functions to maintain a constant identifier, even if they are moved to a new physical address (node address and/or task number). The get function IDs command causes the slave device, or slave device extension, to respond with a list of function ID codes for the tasks currently in existence.

The get function IDs order message carries a dummy data field equal in length to the number of function IDs to be returned. The length field is used to identify the number of bytes (value in length field equals number of bytes plus 7). Carrying the dummy data field allows the slave node to allocate a single buffer that is sufficiently large to receive the order message, as well as return the reply message.

After execution of the get function IDs command, a reply message is returned to the originating task on the master node with 00H in the response field and the function ID codes in the data field. The first

data byte corresponds to task 0, the second data byte to task 1, etc. The number of data bytes returned is always equal to the number sent in the order message.

The assignment of function IDs is summarized in table 6. As an example, consider a slave device that supports up to 8 tasks. Task 0 is RAC, task 1 is a user task with function ID 81H, tasks 2 and 3 are present but have no function ID and tasks 4-7 are not used. For this case, the length field in the order message and reply message is 15. The reply message would have a data field as shown in figure 34.

Table 6. Function ID Code Assignments

FUNCTION	VALUE
No Task	00H
RAC Task	01H
Reserved by Intel	02H-7FH
User Assigned	80H-0FEH
Task with no Function ID	0FFH

TASK NUMBER	MSB					LSB					TASK FUNCTION	
	LENGTH											
	MT	SE	DE	TR	RESERVED (4 BITS)							
	NODE ADDRESS											
	SOURCE TASK					DESTINATION TASK						
	COMMAND/RESPONSE											
	0	01H										
	1	81H										
2	0FFH											
3	0FFH											
4	00H											
5	00H											
6	00H											
7	00H											

RAC
User Function 81H
Task without function ID
Task without function ID
No Task
No Task
No Task
No Task

Figure 34. GET Function ID Reply Example

6.2.4 RAC PROTECT

The RAC protect command allows the remote access functions at a slave node to be enabled or disabled by a task on the master node. When disabled, the RAC function at a slave node only recognizes the remote control commands. On reset, the RAC function is enabled.

The RAC protect order contains a single byte data field and a length field of 8. If the value of this data field is 00H, the RAC function is enabled. If the value in this data field is 01H, the RAC function is disabled. Any other value may produce indeterminate results and should not be used.

After successful completion of a RAC protect command, a reply message is returned to the originating task on the master node with 00H in the response field and the data field unchanged.

6.2.5 RESET SLAVE

The reset slave command initiates a reset at the addressed slave device or slave device extension. The extent of the reset (e.g. software, slave device, slave device extension, entire slave node, etc.) is determined by the implementation. The data field for this command is null and the length field is always 7.

The reset slave command is the only command for which a reply message is not returned upon successful completion. A reply message is not returned in this case since its delivery cannot be predicted while the slave node is in the process of resetting. The only case when a reply message is attempted is in the case of error. This case is discussed in section 6.3 of this specification.

6.2.6 MEMORY COMMANDS

The memory commands allow blocks of data to be moved between the master node and a slave node. Operations include memory upload and memory download. Since these commands share a common format in the message data field, they are presented together.

The general data field format for memory commands is shown in Figure 35. This format includes a 16 bit address pointer and 1 to n bytes of data. The length field in the header is used to specify the actual number of data bytes (e.g. length equal to 9 plus n for n data bytes). The address pointer is used to identify the base address for the data bytes. That is, the pointer is the address associated with data byte 1 and the address of subsequent data bytes is obtained by simply incrementing the pointer.

The address pointer is limited to a 64K address range as this is believed sufficient for the vast majority of BITBUS interconnect applications. In the few cases where a larger address range is needed, it can be created by assigning a location in the I/O or status space as an offset register, thus allowing the memory commands to operate in 64K byte segments or pages.

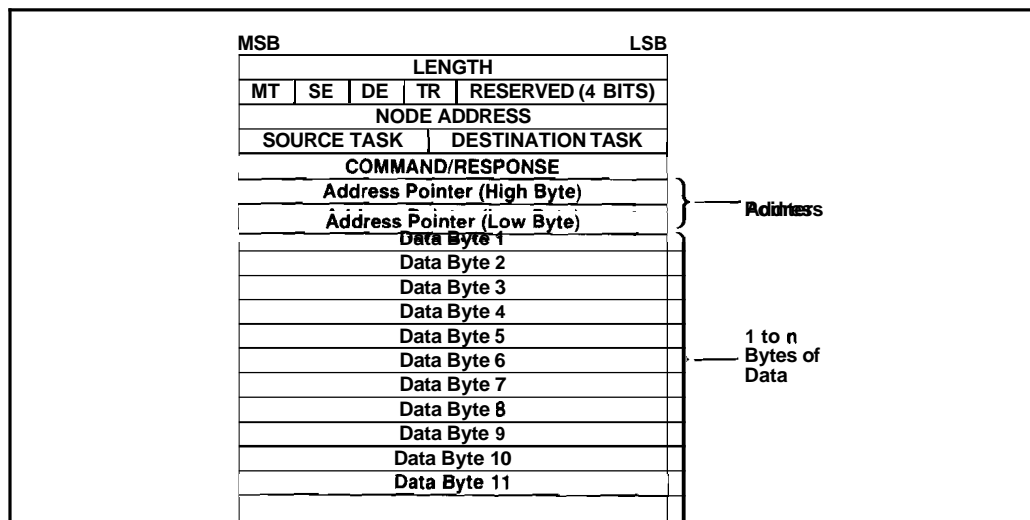


Figure 35. Data Field Format for Memory Commands

6.2.6.1 Memory Upload

The memory upload command causes a slave device, or slave device extension, to read a specified number of data bytes from its local memory and return them in a reply message. The specific number of bytes to be read is identified in the length field of the order message (i.e. length field equals number of bytes to be read plus 9).

The data field in a memory upload order message contains a 16 bit address pointer and n invalid data bytes, where n is the number of bytes requested. The invalid data bytes are carried to simplify buffer allocation at the slave node by allowing a single buffer to be allocated that is sufficient for delivering the order and returning the reply.

At the slave node, the RAC task reads sequentially the specified number of data bytes. The first byte is read from the location specified by the address pointer and subsequent bytes are located by incrementing the address pointer. Upon successful completion, a reply message is returned with an 00H response field and a data field with the original address pointer and the requested data bytes.

6.2.6.2 Memory Download

The memory download command causes a slave device, or slave device extension, to write n bytes of data into its local memory. The specific number of bytes to be written is identified in the length field of the order message (i.e. length field equals number of bytes to be written plus 9). In addition to the data bytes, the data field contains a 16 bit address pointer.

At the slave node, the RAC task writes the data bytes sequentially starting at the location specified by the address pointer. Subsequent addresses are located by incrementing the address pointer for each byte. Upon successful completion, a reply message is returned with an 00H response field and a data field containing the contents of the order message.

6.2.7 I/O COMMANDS

The I/O commands allow the master node to access up to 256 I/O ports on each slave device and each slave device extension. Operations include read 110, write 110, update I/O, OR I/O, AND I/O, and XOR 110. Since these commands all share a common format in the message data field, they are presented together.

The general data field format for I/O commands is shown in figure 36. This format allows a single command to be executed on one or more ports in a single message. The data field is comprised of one or more pairs of bytes. The port address fields identify the I/O ports on which the operation is to be performed. The data byte fields contain the data for the operation or act as place holders for the data that results from an operation.

Upon successful completion of an I/O command, a reply message is returned with an 00H response field and a data field with the same port address fields as the order message and data byte fields as specified by the command. In all cases, the reply message length is equal to that of the order message.

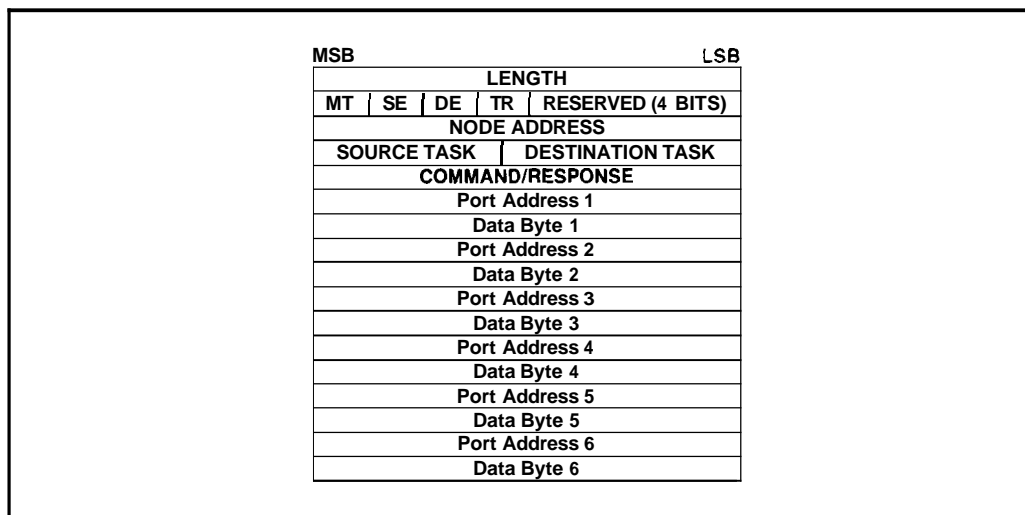


Figure 36. Data Field Format for I/O Commands

6.2.7.1 Read I/O

The read I/O command causes a slave node to read the specified I/O ports. The data byte fields in the order message are undefined. They are carried only to simplify buffer allocation at the slave node. The data byte fields in the reply message contain the data read from the specified ports.

6.2.7.2 Write I/O

The write I/O command causes a slave node to write the data bytes fields to the specified I/O ports. The data byte fields in the order message contain the data to be written. The data byte fields in the reply message are unchanged from the order message.

6.2.7.3 Update I/O

The update I/O command causes a slave node to write the data byte fields to the specified I/O ports, then reread the ports. The data byte fields in the order message contain the data to be written. The data byte fields in the reply message contain the data reread from the I/O ports after the write operation.

6.2.7.4 Logical Operations

The logical operation commands provide bit set, bit reset and bit toggle operations. The data byte fields for these operations contain a mask. The command causes the slave node to read the specified I/O port, perform the specified logical operation with the mask in the data byte field, write the result back to the I/O port, then finally reread the port. This final value is entered in the data byte field for the reply message.

Bit operations are defined to logically set, clear and toggle I/O bits. The electrical levels of these operations depends on the implementation. When no signal inversion exists between the internal CPU state and the I/O pin, the operations of set and clear correspond to the electrical high and low states respectively. These implementations are referred to as "active high." If a signal inversion occurs between the internal CPU state and the I/O pin, the electrical definitions of set and clear are reversed. These implementations are referred to as "active low."

From the logical point of view, bit operations are defined as follows. A bit is set by using an OR I/O command with a one in the bit position to be set. A bit is cleared by using an AND I/O command with a zero in the bit position to be cleared. A bit is toggled by using an XOR I/O command with a one in the bit position to be toggled. Note that it is possible to operate on multiple bits with the same byte in a single operation.

6.2.8 STATUS COMMANDS

The status commands allow the master node to access up to 256 bytes of memory on each slave device and slave device extension. These operations can be used to create a shared data structure between the master and slave nodes. Specific operations include read status and write status. Since these commands share a common format in the message data field, they are presented together.

The general data field format for status commands is shown in figure 37. This format allows a single command to operate on one or more locations with a single message. The data field is comprised of one or more pairs of bytes. The pairs include an address field and a data byte field. The address fields identify the locations at which the operation specified by the command is performed. The data byte field contains the data for the operation or acts as a place holder for the data that results from the operation.

Upon successful completion of a status command, a reply message is returned with an OOH response field and a data field with the same address fields as the order message and data byte fields as specified by the command. In all cases, the reply message length is equal to that of the order message.

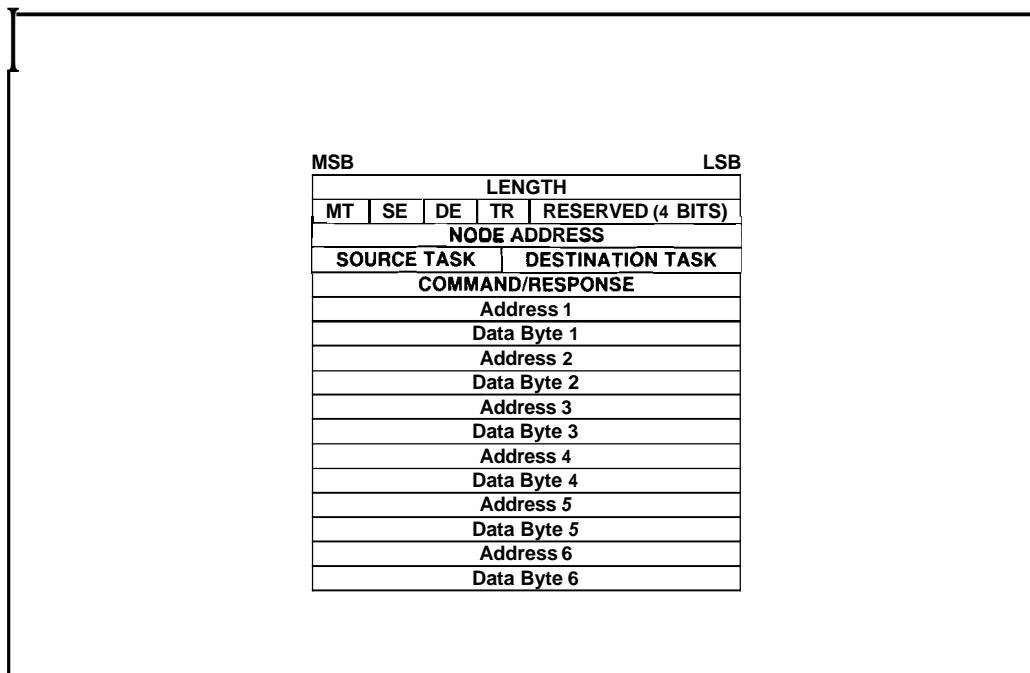


Figure 37. Data Field Format for Status Commands

6.2.8.1 Read Status

The read status command causes a slave node to read the specified memory locations. The data byte fields in the order message are undefined. They are carried only to simplify buffer allocation at the slave node. The data byte fields in the reply message contain the data read from the specified addresses.

6.2.8.2 Write Status

The write status command causes a slave node to write the data byte fields to the specified addresses. The data byte fields in the order message contain the data to be written. The data byte fields in the reply message are unchanged from the order message.

6.3 RAC RESPONSES

In general, RAC order messages expect RAC reply messages. The only cases where reply messages are not returned are for a successfully delivered reset slave order message or under certain error conditions as specified in section 5 (section 5 also specifies how to handle this case). When a reply message is returned, it indicates one of three cases: no error, message protocol error, or RAC error. Reply messages for the no error case are specified for each command in section 6.2. Reply messages for message protocol errors are specified in section 5. This section specifies the reply messages for the various RAC error conditions.

In general, RAC errors generate a reply message similar to message protocol errors. The reply message for RAC errors is simply the order message with the error response inserted in place of the command. the data field is returned intact. Table 7 summarizes the RAC error responses.

6.3.1 NO TASK

No task is generated in response to a delete task command in the case where the task to be deleted does not exist.

Table 7. RAC Error Responses

RESPONSE	VALUE
No Task	80H
Task Overflow	81H
Register Bank Overflow	82H
Duplicate Function ID	83H
No Buffers	84H
RAC Protected	95H
Unknown RAC Command	96H

6.3.2 TASK OVERFLOW

Task overflow is generated in response to a create task command in the case where the slave device or slave device extension cannot support another task. This response indicates that it is necessary to delete a task before another can be created.

6.3.3 REGISTER BANK OVERFLOW

Register Bank Overflow is generated in response to a create task command that cannot be executed due to a lack of register bank resources.

6.3.4 DUPLICATE FUNCTION ID

Duplicate Function ID is generated in response to a create task command in the case where the task to be created has the same function ID (0FFH excepted) as a task currently recognized by the operating system.

6.3.5 NO BUFFERS

No buffers is generated in response to a create task command that cannot be executed due to a lack of memory resources.

6.3.6 RAC PROTECTED

RAC protected is generated in response to any remote access command when the RAC function is disabled. Before the remote access command can be recognized, a RAC protect command must be sent to enable the RAC function.

6.3.7 UNKNOWN RAC COMMAND

Unknown RAC command is generated in response to any RAC command that is not recognized. This response is used for both undefined and unimplemented commands.

7.0 MECHANICAL SPECIFICATION

The BITBUS interconnect provides mechanical specifications for cable connectors and a standard printed board. The cable connectors are used to connect cables to printed boards or cables to other cables. The standard printed board is specified to allow multiple manufacturers to build complementary products that fit into a consistent packaging scheme.

7.1 CONNECTOR SPECIFICATIONS

The BITBUS interconnect requires specification of two standard connectors: one for cable to printed board connections and one for cable to cable connections. Both connectors support the same signals. Four pins are used for the two differential signal pairs specified in section 2 of this

specification for the BITBUS interconnect. Four additional pins are provided for power distribution to low power nodes. Finally a high impedance ground (100 ohms to ground) is provided as specified in the RS485 specification (not required in all implementations). Figure 38 and 39 show the two connectors and table 8 lists the pin assignments.

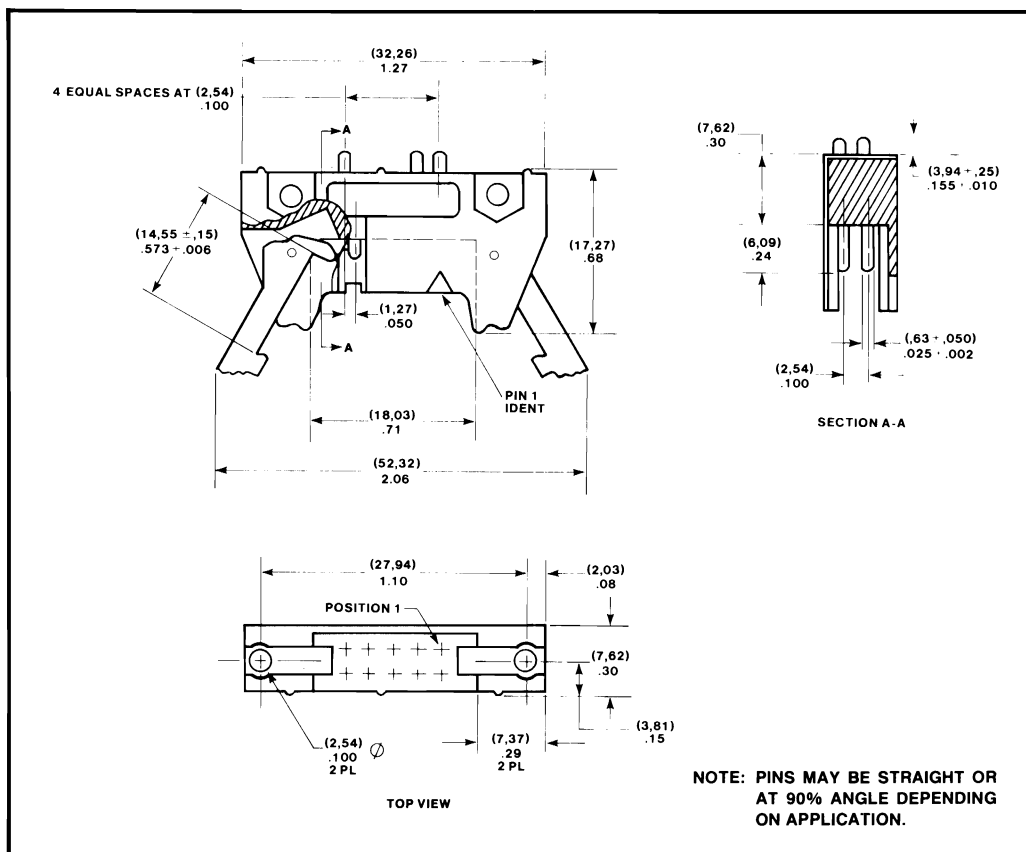


Figure 38. Printed Board Connector

Table 8. Connector Pin Assignments

SIGNAL	PRINTED BOARD CONNECTOR PIN #	CABLE CONNECTOR PIN #
+ 12V \pm 5%	1	1
+ 12V \pm 5%	2	6
GND	3	2
GND	4	7
DATA *	5	3
DATA	6	8
DCLK */RTS *	7	4
DCLK/RTS	8	9
RGND (100 ohms to GND)	9,10	5

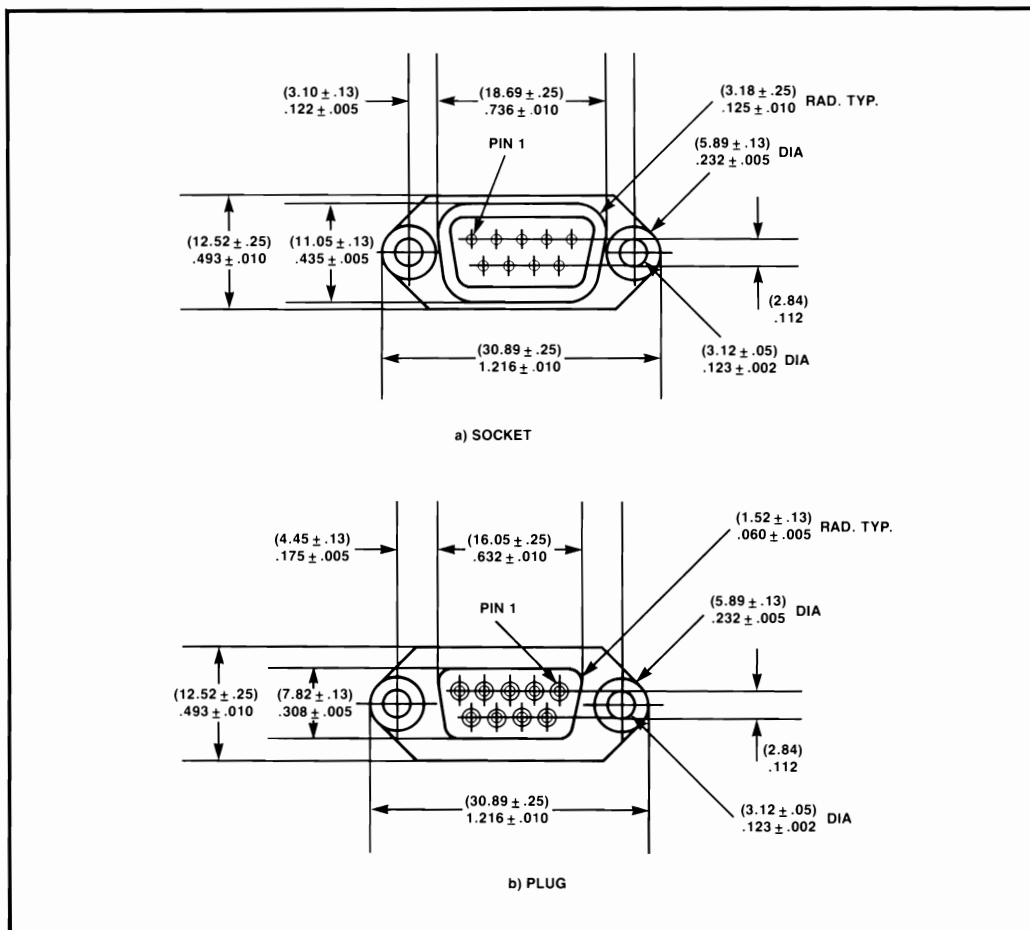


Figure 39. Cable to Cable Connector

The standard cable to printed board connector is a 10 pin latching pin and socket type with strain reliefs. This connector can support either flat cable or discrete wire connection.

The standard cable to cable connector is a 9 pin D-subminiature type. Versions of this connector are also available for flat cable or discrete wire connection.

7.2 I/O BOARD SPECIFICATION

The BITBUS I/O board is designed to be compatible with the single-high Multibus II board form factor. This definition is based on the International Electrotechnical Commission (IEC) standards. This section provides an overview of the mechanical dimensions of this board and presents standard pin assignments. For further details (e.g. board spacing, component height, ejector tabs, front panels, etc.) refer to the Mechanical Specification in the Multibus II Bus Architecture Specification.

7.2.1 BOARD DIMENSIONS

Figure 40 shows the board dimensions for the standard BITBUS I/O board. This board is a standard single-high, 220mm deep form factor.



7.2.2 CONNECTORS

The BITBUS I/O boards use two piece, 64-pin connectors. Figure 41 and 42 show the dimensional specification for the connectors. The right angle connectors on the printed board are IEC standard 603-2-IEC-C064-M; the receptacle connectors are IEC standard 603-2-IEC-C064-F. Figure 43 shows the relationship between connectors and boards in a subrack. Note that compatible receptacle connectors are available for flat cable, discrete wire or wire wrap connections in addition to the backplane version shown.

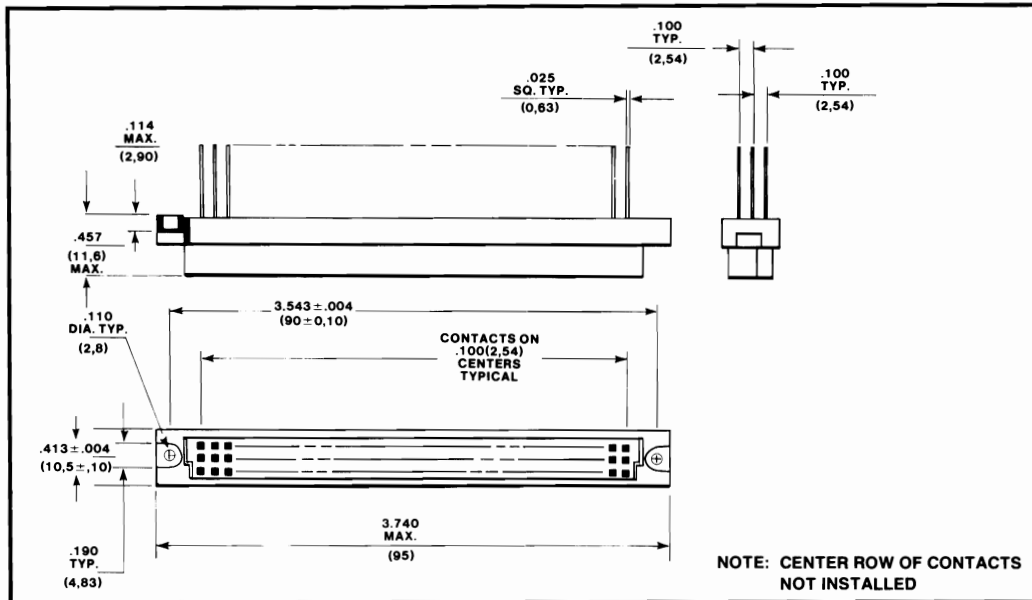


Figure 41. Board Connector

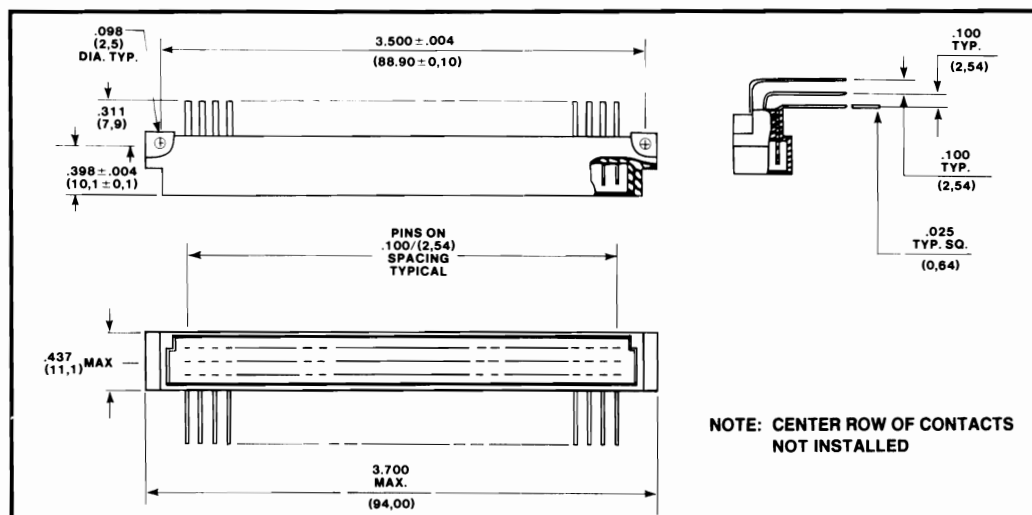
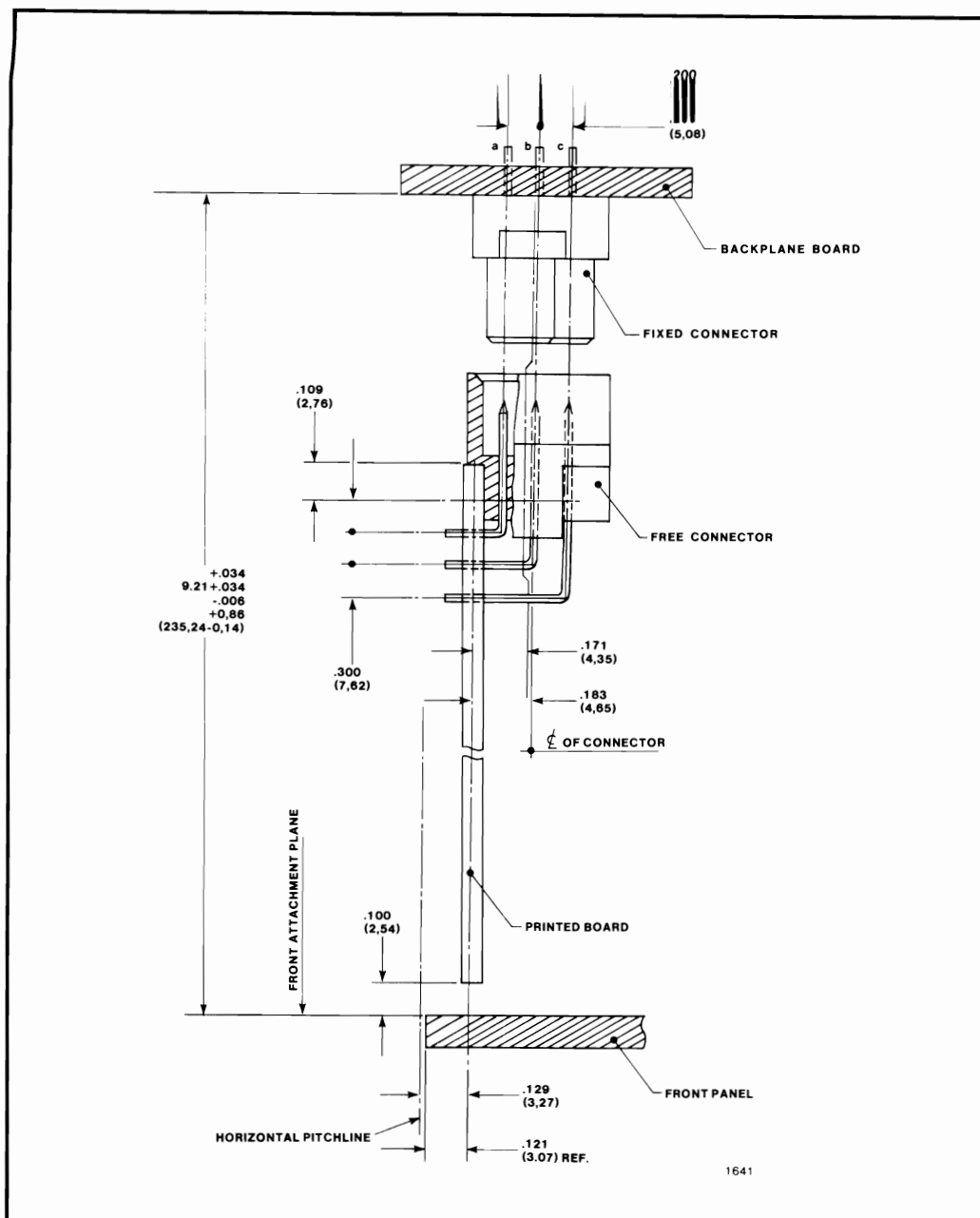


Figure 42. Backplane Connector



7.2.3 PIN ASSIGNMENTS

The standard BITBUS I/O board defines standard pin assignment for the 64 pin connector as shown in table 9. This pin assignment defines pins for the BITBUS interconnect as defined in section 2 of this specification, power and ground (1 amp per pin), I/O signals and signal ground returns. The I/O signal lines are defined as required by each implementation.

Table 9.110 BoardPin Assignment

PIN#	SIGNAL	PIN #	SIGNAL
1a	GND	1c	GND
2a	+5V	2c	+5V
3a	DATA	3c	DATA *
4a	DLCK/RTS	4c	DLCK*/RTS*
5a		5c	
6a		6c	
7a		7c	
8a		8c	
9a		9c	
10a		10c	
11a		11c	
12a		12c	
13a		13c	
14a		14c	
15a		15c	
16a		16c	
17a		17c	
18a		18c	
19a		19c	
20a		20c	
21a		21c	
22a		22c	
23a		23c	
24a		24c	
25a		25c	
26a		26c	
27a		27c	
28a		28c	
29a		29c	
30a	+12V	30c	-12V
31a	+5V	31c	+5V
32a	GND	32c	GND

* = user defined I/O

8.0 LEVELS OF COMPLIANCE

The **BITBUS** interconnect specification contains many features that are required for compatibility, and other features that are only required for additional capabilities. This section clarifies the various levels of compliance that may be supported by various implementations of the **BITBUS** interconnect. This information is included to allow use of the **BITBUS** interconnect products of varying capabilities manufactured by diverse vendors.

8.1 CONCEPT OF LEVEL OF COMPLIANCE

The concept of level of compliance is provided to guarantee compatibility of **BITBUS** interface products that support the specification at different levels. The basic rule is that all products are required

to support a minimum level of the specification. This results in a minimal level at which all BITBUS interconnect products can interact. Furthermore, enhancements beyond this minimal level are controlled. This results in a guaranteed compatibility between a group of products that all support the same enhanced capabilities.

8.2 COMPLIANCE LEVELS

This section presents the various levels of compliance for each level of the specification. Each capability supported beyond the minimum shall be documented in the product data sheet.

8.2.1 ELECTRICAL INTERFACE

The electrical interface supports variability in bit rates, load characteristics, and repeater characteristics. The bit rates specified in section 2 include 2.4 Mbits/sec synchronous and 375 kbits/sec and 62.5 kbits/sec self clocked. Of these only 375 kbits/sec is required. The others may be optionally supported. The load for a BITBUS node shall be specified in terms of the standard load defined in section 3.2.1. Repeaters are specified in terms of the standard repeater delays defined in section 3.3.5.

8.2.2 DATA LINK PROTOCOL

The data link protocol has no optional features. The full protocol shall be supported for compatibility.

8.2.3 MESSAGE PROTOCOL

The message protocol supports variability in message length. All implementations shall support the standard 5 byte header and 13 byte data field. Implementations may optionally support larger data fields.

8.2.4 REMOTE ACCESS AND CONTROL

Support of the remote access and control function is completely optional. The only requirement is that supported commands shall meet the specification.

8.2.5 MECHANICAL

The mechanical specification contains the option of supporting the standard I/O board specification. As with RAC, the only requirement is that if supported, it shall meet the specification.



UNITED STATES, Intel Corporation
3065 Bowers Ave., Santa Clara, CA 95051
Tel: (408) 765-8080

JAPAN, Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi, Ibaraki, 300-26
Tel: 029747-8511

FRANCE, Intel Corporation S.a.r.l.
1, Rue Edison, BP 303, 78054 Saint-Quentin-en-Yvelines Cedex
Tel: (33) 1-30 57 70 00

UNITED KINGDOM, Intel Corporation (U.K.) Ltd.
Pipers Way, Swindon, Wiltshire, England SN3 1RJ
Tel: (0793) 696000

WEST GERMANY, Intel Semiconductor GmbH
Seidlstrasse 27, D-8000 Muenchen 2
Tel: (89) 53891

HONG KONG, Intel Semiconductor Ltd.
10/F East Tower, Bond Center, Queensway, Central
Tel: (5) 8444-555

CANADA, Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8
Tel: (416) 675-2105